TU Clausthal

COHERENT AUGMENTED REALITY RENDERING FOR MOBILE AND NON-MOBILE DEVICES

DOCTORAL THESIS
(DISSERTATION)

TO BE AWARDED THE DEGREE

DOCTOR RERUM NATURALIUM (DR. RER. NAT.)

SUBMITTED BY
KAI ROHMER
FROM NAUMBURG (SAALE)
(place of birth)

APPROVED BY THE FACULTY OF
MATHEMATICS/COMPUTER SCIENCE AND MECHANICAL ENGINEERING,
CLAUSTHAL UNIVERSITY OF TECHNOLOGY,

DATE OF ORAL EXAMINATION
MAY 25, 2018

DEAN
Prof. Dr.-Ing. Volker Wesling

CHAIRPERSON OF THE BOARD OF EXAMINERS
Prof. Dr.-Ing. Michael Prilla

SUPERVISING TUTOR
Prof. Dr. rer. nat. Thorsten Grosch

REVIEWER
Univ.-Prof. Dipl.-Ing. Dr. techn. Dieter Schmalstieg,
Graz University of Technology

ACKNOWLEDGMENTS

This thesis is product of my work in the Computational Visualistics Group at the Department of Simulation and Graphics of the Otto-von-Guericke-University Magdeburg and later as part of the Graphical Data Processing and Multimedia Group at Clausthal University of Technology, both headed by my advisor Thorsten Grosch. Therefore, it's you Thorsten I want to start expressing my gratitude to. I'm sincerely thankful for the many opportunities you have offered me, for the support, guidance, and freedom you gave me, and for everything you taught me in all those years.

I'm very grateful to a lot of people I met during the time in Magdeburg. Thank you Tobi, my fellow student, colleague and friend, for the countless discussions and the wonderful time we spent working on inspiring and fun projects. Thanks Holger and Christian for introducing me to the exciting and highly motivating field of computer graphics. Along with Maik, Janick, Timo, Dirk, Tim, Thomas, John and Anke, it was always a real pleasure. I loved being (sort of) part of your group and I'm grateful for the very pleasant but also highly productive atmosphere, the many things I learned from you, and of course the time we spent during and after lunch. Thank you all!

Even though my time in Clausthal was much shorter, I got the chance to meet some great people I also want to show my thanks to. Thank you Andreas, for being a mentor, colleague, running partner and friend. You made life on that rainy hill very enjoyable. The same goes to Feng, Dima, and Gerrit. And also thanks for the coffee breaks!

I would like to include a special note of thanks to my other roommates Sophie and Johannes. Sharing an office with you has always been a pleasure.

And of course, I thank my family and friends for the unwavering support before, after, and throughout all my studies at the university. You always have my back. Thank you so much! Finally and above all, I wish to thank my loving and supportive sweetheart Franzi for her endless motivation, inspiration, confidence and patience. You make sure that I still have a life besides work and I'm extremely happy to be able to enjoy it together with you!

ABSTRACT

The subject of this thesis is the interactive and visually coherent augmentation of live camera images. Considering the dynamic environment, the goal is to embed virtual objects seamlessly into the given image data. For this, it is necessary to precisely acquire the geometry and material properties of the real surfaces in order to subsequently perform illumination simulations in this reconstruction.

During the preparation and creation of visual effects in the film industry, this real-world information already plays a significant role and enables a believable combination of virtual and real elements. Since capturing the information is not trivial and a correct lighting simulation is time consuming, a simplified visualization of objects without regard to the real environment is used to achieve interactive display rates in case of augmented reality. In many applications, such as evaluating virtual design prototypes, this simplified visualization is insufficient. The most correct rendering possible, that also considers the real surrounding of the virtual objects, is necessary to serve as a basis for decision-making.

Accordingly, this dissertation explores new approaches to capture the real-world environment of virtual objects and, based on this, to realize a coherent interactive visualization. Particular attention is paid to direct and strong indirect light sources, which have a significant influence on the appearance of the virtual objects. The current state of the art is based on differential rendering, in which global light transport simulations are conducted to determine the influence of virtual objects on the real environment. The methods developed in the scope of this dissertation are based on that technique, too. However, they are designed to produce high-quality yet highly performant augmentations that are suitable for interactive use on mobile devices. Because of the development and dissemination of mobile devices, which provide many sensors and interaction possibilities, they became an extremely relevant platform for augmented reality solutions in various areas of our everyday life as well as for professional usage. The methods presented in this dissertation seize the potential of the mobile platform and use it in a way that no other previous publication has demonstrated.

The core of the presented work are two physically-based augmented reality rendering frameworks: A distributed system that outsources the acquisition of the environment and the computationally expensive extraction of light sources to a stationary PC. The resulting compact parametrization of a lightweight illumination model is constantly updated and transmitted to mobile devices, which use their own computing capacity for an interactive and individual presentation to the user. The second system is based on a mobile device equipped with a depth sensor. It does not require any additional hardware. The environment is recorded as a three-dimensional point cloud, which is then used as input for light simulation methods. The adaptation of GPU-based Monte Carlo rendering provides a trade-off between quality and performance, and thus interactive display on mobile devices as well as a photorealistic rendering, that is otherwise known only from offline methods. Furthermore, a method for estimating unknown color transformations of cameras is presented, which is used during the scene acquisition to measure surface radiance with high dynamic range. They are also used as a color adjustment between virtual and real objects, making the boundaries harder to perceive and the augmentation more seamless.

This dissertation introduces new methods that improve augmented reality rendering on mobile devices in terms of quality and performance. Thus, they improve the current state of the art and offer new possibilities for applications in many fields in which interactive and coherent visualization of virtual elements is of great importance, e.g., for the visualization of design prototypes, architecture, interior design, cultural heritage as well as in museums and exhibitions.

ZUSAMMENFASSUNG

Gegenstand dieser Dissertation ist die interaktive und visuell kohärente Erweiterung von Live Kamerabildern um virtuelle Objekte. Unter Berücksichtigung der teils dynamischen Umgebung, besteht das Ziel darin, die hinzugefügten Elemente möglichst nahtlos in die gegebenen Bilddaten einzubetten. Hierfür ist es notwendig, die Geometrie und die Materialeigenschaften der realen Oberflächen genau zu erfassen, um anschließend Beleuchtungssimulationen durchzuführen.

Während der Vorbereitung und der Erstellung von visuellen Effekten in der Filmindustrie spielen diese realweltlichen Informationen bereits eine bedeutende Rolle und ermöglichen eine glaubhafte Kombination aus virtuellen und realen Elementen. Da das Erfassen der Informationen alles andere als trivial und eine korrekte Beleuchtungssimulation zeitaufwendig ist, wird im Falle von Augmented Reality häufig auf eine stark vereinfachte Visualisierung von Objekten ohne Berücksichtigung der realen Umgebung zurückgegriffen. Dies ermöglicht das Erreichen der zwingend erforderlichen interaktiven Darstellungsraten. In vielen Anwendungsfällen, wie z.B. dem Evaluieren von virtuellen Design Prototypen, ist diese vereinfachte Visualisierung ungenügend. Eine möglichst korrekte Darstellung der virtuellen Objekte in der aktuellen realen Umgebung ist notwendig, um als Entscheidungsgrundlage zu dienen.

Dementsprechend untersucht diese Dissertation neue Ansätze, um das realweltliche Umfeld von virtuellen Objekten zu erfassen und darauf basierend, eine kohärente interaktive Darstellung zu realisieren. Besonderes Augenmerk liegt dabei auf den direkten und starken indirekten Lichtquellen, die einen wesentlichen Einfluss auf das Erscheinungsbild der virtuellen Objekte haben. Eine Beschreibung der Lage und Intensität dieser Lichtquellen ist unabdingbar, um eine korrekte Verschattung zwischen virtuellen und realen Elementen der Szene zu berechnen. Während viele aktuell genutzte Verfahren annehmen, dass sich die Lichtquellen in unendlich weiter Entfernung befinden, ist es Ziel dieser Arbeit ohne diese Annahme auszukommen und damit auch eine korrekte Nahfeldbeleuchtung zu ermöglichen. Der aktuelle Stand der Technik basiert auf Differenziellem Rendering, in dem globale Beleuchtungssimulationen durchgeführt werden, um den Einfluss von virtuellen Objekten auf die reale Umgebung zu bestimmen. Dieser Einfluss beinhaltet neben Verdeckung und Verschattung auch indirekte Beleuchtung, die durch die Reflexion von Licht zwischen realen und virtuellen Elementen entsteht. Auch die im Rahmen der vorliegenden Arbeit entwickelten Verfahren beruhen auf dieser Technik. Sie zielen aber darauf ab, qualitativ hochwertige und gleichzeitig performante Bildsynthesen zu erzeugen, die für den interaktiven Einsatz auf mobilen Geräten geeignet sind. Durch die Entwicklung und Verbreitung von mobilen Endgeräten mit einer Fülle an Sensoren und Interaktionsmöglichkeiten, stellen diese eine äußerst relevante Plattform für die Augmented Reality Lösungen in verschiedensten Bereichen des alltäglichen Lebens und für den professionellen Einsatz dar. Die in dieser Dissertation vorgestellten Beiträge greifen das Potenzial der mobilen Plattform auf und nutzen es in einer Weise, die in bisher keiner anderen Publikation demonstriert wurde.

Kern der vorliegenden Arbeit sind zwei physikalisch fundierte AR Rendering Frameworks: Ein verteiltes System, welches das Erfassen der Umgebung und die rechenintensive Extraktion von Lichtquellen auf einem stationären Computer auslagert. Die resultierende kompakte Parametrisierung eines leichtgewichtigen Beleuchtungsmodelles wird permanent aktualisiert und an mobile Geräte übertragen, die die eigenen Rechenkapazitäten für eine interaktive und individuelle Darstellung von virtuellen Objekten nutzen. Das zweite System basiert auf einem mobilen Gerät mit Tiefensensor und kommt ohne zusätzliche Hardware aus.

In einer Initialisierungsphase wird die Umgebung als dreidimensionale Punktwolke mit Oberflächennormale und Leuchtdichte für jeden gemessenen Punkt aufgenommen. Diese Rekonstruktion wird im Anschluss als Eingabe für verschiedene Lichtsimulationsverfahren genutzt. Die Adaption von GPU-basiertem Monte Carlo Rendering für die Anwendung in Augmented Reality Szenarien ermöglicht einen Trade-off zwischen Qualität und Geschwindigkeit und damit sowohl interaktive Darstellung auf mobilen Geräten, als auch eine photorealisitische Darstellung, die sonst nur aus Offline-Verfahren bekannt ist. Des Weiteren wird ein Verfahren zur Schätzung von unbekannten Farbtransformationen von Kameras vorgestellt, was während der Aufnahme genutzt wird, um Leuchtdichten mit hohem Dynamikbereich zu messen. Während der Darstellung, wird die geschätzte Transformation als Farbabgleich zwischen virtuellen und realen Objekten genutzt, wodurch beide Teile visuell besser verschmelzen und die Grenzen damit schwerer wahrnehmbar werden.

Diese Dissertation stellt damit neue Verfahren vor, die Augmented Reality Darstellungen auf mobilen Endgeräten visuell als auch aus Sicht der Performanz verbessern. Sie ergänzen damit den bisherigen Stand der Technik und bieten neue Möglichkeiten für Anwendungen in vielen Bereichen in denen eine interaktive und kohärente Visualisierung von virtuellen Elementen von großer Bedeutung ist, z.B. Visualisierung von Design Prototypen, Architektur, Inneneinrichtung, Denkmalpflege sowie in Museen und Ausstellungen.

CONTENTS

NOMENCLATURE

p	Scalar value
$F()$	Scalar-valued function
$F^{-1}()$	Inverse of a scalar-valued function
$\mathbf{x}$	Vector (column-wise) in n-D space
$\mathbf{x}^T$	Transpose of a vector or matrix
$\mathbf{x}^T\mathbf{y}$	Dot product (in matrix notation)
A	Matrix
A^{-1}	Inverse of a matrix
I	Identity matrix
$\mathcal{R}$	Set
$\boldsymbol{\omega}$	A solid angle $[sr]$ or the direction of an infinitesimal solid angle
$\overline{\cos}\,\theta$	The cosine of the angle θ clamped to $[0, 1]$
$\int_{\mathcal{H}_+}$	Spherical integral over the domain of the upward hemisphere
$\int_{\mathcal{H}_-}$	Spherical integral over the domain of the downward hemisphere
$\int_{\mathcal{H}_\pm}$	Spherical integral over the domain of both hemispheres
$\mathbf{N}(\mathbf{x})$	Surface normal at position $\mathbf{x}$
$\frac{\partial f}{\partial t}$	Partial derivative of a (multivariate) function f

ACRONYMS

AO	Ambient Occlusion
API	Application Programming Interface
AR	Augmented Reality
BPT	Bidirectional Path Tracing
BRDF	Bidirectional Reflectance Distribution Function
BSDF	Bidirectional Scattering Distribution Function
BSSRDF	Bidirectional Surface Scattering Reflectance Distribution Function
BTDF	Bidirectional Transmittance Distribution Function
BTF	Bidirectional Texture Function
BVH	Bounding Volume Hierarchy
CAD	Computer-aided Design
CDF	Cumulative Density Function
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DCC	Digital Content Creation
DI	Distance Impostor
DIT	Distance Impostor Tracing
DLPV	Delta Light Propagation Volume
DVCT	Delta Voxel Cone Tracing
EM	Environment Mapping
FAST	Features from Accelerated Segment Test
GI	Global Illumination
GPU	Graphics Processing Unit
GPGPU	General Purpose Computation on the Graphics Processing Unit
HDR	High Dynamical Range
HSV	Hue-Saturation-Value
IBL	Image-based Lighting
ICP	Iterative Closest Point
ILF	Incident Light Field
IR	Instant Radiosity
ISM	Imperfect Shadow Maps
iTMO	Inverse Tone Mapping Operator
LCD	Liquid Crystal Display
LDR	Low Dynamical Range
LoD	Level of Detail
LPV	Light Propagation Volume
MR	Mixed Reality
MLT	Metropolis Light Transport
NPC	Normalized Projection Coordinates

OST	Optical See-Through
PBR	Physically-based Rendering
PDF	Probability Density Functions
PM	Photon Mapping
PRT	Precomputed Radiance Transfer
RSM	Reflective Shadow Map
SAR	Spatial Augmented Reality
SH	Spherical Harmonics
SLAM	Simultaneous Localization and Mapping
SSDO	Screen-Space Directional Occlusion
SURF	Speeded Up Robust Features
SVBRDF	Spatially-Varying Bidirectional Reflectance Distribution Function
SVD	Singular Value Decomposition
TLL	Tiled Light List
TMO	Tone Mapping Operator
VAL	Virtual Area Light
VCM	Vertex Connection and Merging
VCT	Voxel Cone Tracing
VPL	Virtual Point Light
VSL	Virtual Spherical Light
VR	Virtual Reality
VST	Video See-Through

1

INTRODUCTION

The creation of immersive artificial worlds is part of human history for a long time, starting with stories and tales that where passed down orally, over paintings showing events in the ancient past or the far future, and literature of any kind that fascinates and absorbs the reader. Mainly driven by newer media forms, such as movies and games, virtual environments based on computer generated three-dimensional content became more and more popular in the last decades. With the technology evolving, virtual environments are getting increasingly realistic – at least in terms of visuals and sometimes audio. The degree of immersion in such artificial worlds is currently seeing a new boost by the recurring popularity of tracked head-mounted displays that allow the users to change their vantage point as they would do in the real world. The term Virtual Reality (VR) however, stands for a broader concept than just spectating a realistic virtual world:

> *VR environment is one in which the participant-observer is totally immersed in, and able to interact with, a completely synthetic world. Such a world may mimic the properties of some real-world environments, either existing or fictional; however, it can also exceed the bounds of physical reality by creating a world in which the physical laws ordinarily governing space, time, mechanics, material properties, etc. no longer hold.*
>
> — Milgram and Kishino [MK94]

With increasing capabilities of mobile devices, these virtual worlds and their unbound possibilities can be reached at any time from any place. However, they are detached from our physical surrounding and thereby detached from our daily life [SH16]. To be able to use this technology to ease real-world problems, to provide meaningful information about complex tasks or simply to enhance collaboration and the transfer of knowledge concerning aspects in the local environment, a connection between virtual elements and our physical surrounding is most desirable. This kind of connection is the intrinsic motivation behind Augmented Reality (AR).

> *Whereas virtual reality (VR) places a user inside a completely computer-generated environment, augmented reality (AR) aims to present information that is directly registered to the physical environment. (AR) goes beyond mobile computing in that it bridges the gap between virtual world and real world, both spatially and cognitively. With (AR), the digital information appears to become part of the real world, at least in the user's perception.*
>
> — Schmalstieg and Höllerer [SH16]

Creating this connection between the real and the virtual world is the great goal and some people might argue that this is done for a long time and it is well known from special effects in movies, where computer generated content is added to real-world footage [SH16]. But this is not true when considering AR in the most widely accepted interpretation. According to AZUMA, three criteria have to be fulfilled: the combination of real and virtual, interactive in real-time and registered in 3D [Azu97]. This allows AR to be used as interaction technique, anchored in the real world and independent from the specific technical realization. Ideally, we will have access to all kind of information without the need of any buttons or screens. Instead, AR will allow to interact with virtual objects as we do with real ones in a very natural and intuitive way.

According to AZUMA's definition, special effects in movies lack the important aspect of interactivity. By incorporating techniques developed in the game industry, movie companies are working on interactive experiences, but mainly in the context of VR. However, video sequences with special effects that took days to compute single frames several years ago, are getting more and more feasible in real-time with today's technology, so eventually, producing immersive experiences in real-time is only a matter of time. The big challenge in AR is the augmentation of our personal environment that is not directed by movie producers. It requires sophisticated techniques to reliably understand the individual, complex and more or less unknown real-world environment we are currently working and living in. Therefore, it is an interdisciplinary challenge, that involves not only computer graphics and computer vision or computer science in general, but also expertise from the individual field of application (see Section 1.1).

Also because of the role of movies and games in the development process, the focus is mainly on the visual (and aural) aspects. Despite the fact, that AR is not restricted to visual augmentations – nor are visual super-impositions required at all, this thesis concentrates on the visual appearance of virtual objects in an augmented world. More precisely, the presented work focuses on visual coherence in augmented reality. For many years, visual coherence is one of the research topics in focus of the computer graphics community. With the goal of photorealistic augmentation, various methods have been developed to insert virtual objects into a view of the real world so that they blend in seamlessly. Sophisticated methods exist, that allow for plausible global illumination light transport simulations – the foundation of photorealistic rendering – at interactive rates, but only on non-mobile platforms. Mobile devices like smartphones and tablet PCs, that became part of our everyday life, have all properties required for AR though. They are portable and can be used wherever we need them, they have sensors that allow for registration in our 3D world and they can display the virtual elements in the live camera stream that shows the real environment. Using the integrated camera, such a device can act like a "porthole window" into an augmented real world as suggested by FITZMAURICE *et al.* [FZC93].

In the past, when virtual objects are inserted, their appearance is often inconsistent with the real environment, especially in mobile AR. To achieve coherent augmentations, there are three major topics we need to deal with: geometric registration, photometric registration and camera simulation [SH16]. Geometric registration of the virtual content and the real world is the foundation of coherent AR applications. It involves a pose estimation of the camera in the real world and its projection parameters. Given these parameters, we

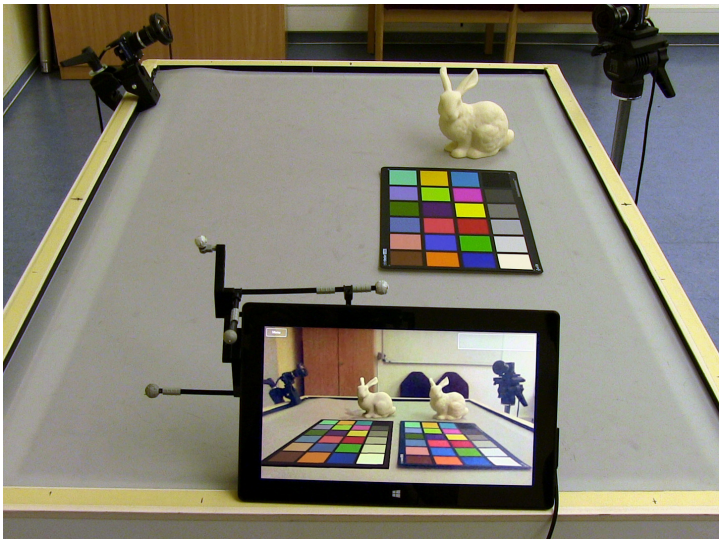


Figure 1.1: Demo Scenario for Distributed Near-Field Illumination
The tablet camera shows the real world, augmented by virtual objects with consistent illumination, displayed at 27 Hz. By using a tracked device, the user can move the tablet freely in the augmented real world and interact with the inserted objects (see Chapter 5).

can automatically provide several important visual cues while rendering virtual objects, e.g., relative size of objects and perspective. This allows us to focus on occlusions, shadows and shading to achieve a seamless blend of virtual and real objects. Camera simulation takes the physical behavior of the camera into account. As cameras in mobile devices cannot be compared to professional DSLR cameras – or even the human eye – in terms of quality, mainly because of size and price, we need to deal with various kinds of artifacts that influence our view into the real world and thereby the augmented world presented to the user. While this issue is addressed in Chapter 7, the main topic of this thesis is the aspect of photometric registration: It describes the interaction of light between the real world and the virtual augmentations. This involves the acquisition of the real environment including geometry, materials and light – ideally, the plenoptic function introduced by Adelson and Bergen [AB91]. Based on the captured data, light simulation approaches can be used to compute the appearance of the virtual objects in the real environment and their virtual influence back on the real scene. Figure 1.1 contains a result of the approach described in Chapter 5. The virtual objects are illuminated by the acquired environment light and thereby look very similar to their real counterparts. From the point of view of classical computer graphics, this is the main problem on the way of achieving a seamless blending of the perceived appearance of real and virtual objects. Especially, in dynamic environments this results in a very complex and challenging task:

> *Estimating real-world lighting and applying it to virtual objects is a key element of visually coherent rendering in AR. In the real world, shadows change, as people and objects move, and lighting changes, as lamps are switched on and off.*
>
> — Gruber *et al.* [GVS15]

So far, a consistent illumination that handles both dynamic scenes and dynamic lighting conditions eluded the mobile platforms and several applications would benefit from such mobile systems. The main contributions of this thesis, presented in Chapters 5 to 7, include several techniques that seek to fill this gap by providing high-quality mobile AR at interactive performance.

1.1 APPLICATION AREAS

Possible applications for AR are nearly everywhere. Depending on the particular use case radiometric coherent rendering of virtual objects is not always necessary as many scenarios benefit from stylized visualizations. Common examples are working instructions and step-by-step guides that help users to accomplish their tasks (see Figure 1.2a). This includes maintenance and stan-

(a) Instructions (b) Process Visualization (c) Construction Plans

(d) Navigation (e) Translation (f) Head-up Displays

(g) Enhancing Print Media (h) Interactive 3D Exploration

(i) Interactive 3D Anatomy (j) Therapy (k) Overlaying Scans

Figure 1.2: General Applications for Augmented Reality
There are various applications in the industry as well as in the educational and medical sector, but also related to traveling and sports. Image courtesy of (a) VR Safety Limited, (b) Klimant *et al.* [KKS17], (c) Scottish Construction Now, (d) Daimler AG, (e) Quest Visual, Inc, (f) Recon Instruments, (g) NMY Mixed-Reality Communication GmbH, (h) Microsoft, (i), Microsoft (j) Juan *et al.* [Jua+05] and (k) TU München, respectively.

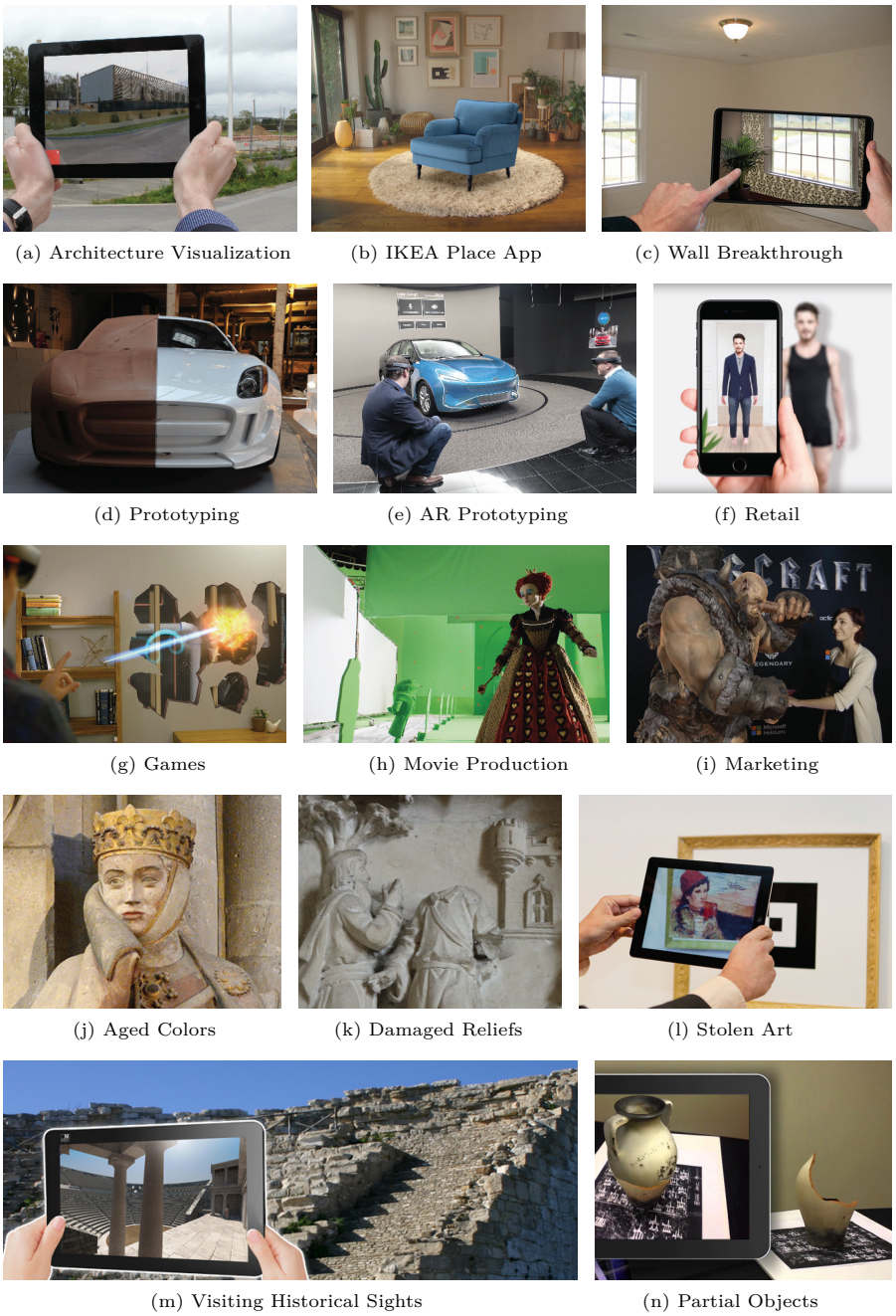
dard repair procedures but also instructions in manufacturing. Besides the guidance and training of users or workers, AR can also be used for in-place visualizations of processes happening inside of machines and devices [Kol+17] (see Figure 1.2b). Similar to that, we can use in-place visualization of existing or future water conduits and power supply lines to support workers and planners during the construction or maintenance of buildings (see Figure 1.2c).

There are various applications for travelers, including navigation by displaying lines and arrows directly on the street (see Figure 1.2d), labels that show the location of the next cafés around you and in-place translations of text that is written in a language you do not speak (see Figure 1.2e). Related techniques are also relevant for sports, where player names can be annotated or lines, distances, trajectories and any other information useful to spectators, can be displayed while the game is running. AR can also be useful to athletes themselves when considering head-up displays for cyclist or runners for instance (see Figure 1.2f).

While these examples do not necessarily require coherent rendering, other areas can benefit from it. In addition, there are applications where a coherent appearance is critical to success, and there are also cases in between. One of them is the field of education and training. The classical transfer of knowledge with the help of texts and images can be supplemented by 3D animations and interactive visualizations (see Figure 1.2g). Compared to images or videos, the object of interest can be observed from arbitrary angles and manipulations by the user can help to discover functionality by trail and error without the risk of breaking anything (see Figure 1.2h). For some professionals, like surgeons, a more realistic visualization during training might help to prepare better for future real-world scenarios, which starts with anatomy classes (see Figure 1.2i). There is also potential for the usage of AR in the treatment of psychological disorders [Jua+05]. One example is the treatment of arachnophobia by confronting patients with their fear in a controlled situation that can be adapted to the patients progress and stopped at any time (see Figure 1.2j).

Interesting scenarios also arise with the demographic change in western societies. AR can help in the sector of informal care, by supporting family members that take care of their effected relatives at home by following predefined or remote instructions from experts [JP16]. Other applications in the field of medicine include overlaying medical data from various scans directly on the patients body (see Figure 1.2k). This can be used for illustration purposes while informing the patients about their condition or during surgery, so that the surgeon can keep his gaze on the instruments rather than turning towards monitors.

We primarily aim for applications which require correct illumination at every location in the scene, and a perceptively plausible lighting is probably not sufficient. Such applications can be found, for example, in interior design and architecture, by presenting convincing in-place visualization of the designer's vision (see Figure 1.3a). We can also support private users, who want to refurnish their homes and would like to know how the sideboard or sofa looks in their own living room (see Figure 1.3b). You can find several apps in the stores that allow this already, but they usually lack coherent illumination. In the field of coherent AR rendering, we are not limited to reproduce the real-world light condition for shading virtual objects. Instead, we also aim for changing real-world materials, adding new light sources like additional lamps, but also supporting the planning of a wall breakthrough for new a window for instance (see Figure 1.3c). Without adding any new furniture, these examples



(a) Architecture Visualization (b) IKEA Place App (c) Wall Breakthrough

(d) Prototyping (e) AR Prototyping (f) Retail

(g) Games (h) Movie Production (i) Marketing

(j) Aged Colors (k) Damaged Reliefs (l) Stolen Art

(m) Visiting Historical Sights (n) Partial Objects

Figure 1.3: Applications for Coherent Augmented Reality

Photorealistic AR can enhance applications in architecture, home furnishing and retail as well as the prototyping for design and development. Marketing, movie productions and games can also benefit from improved coherence. Furthermore, new possibilities in the field of cultural heritage will emerge along with new forms of presentations in museums and exhibitions. Image courtesy of (a) Urbasee, (b) Ikea, (d) Jaguar, (e) Microsoft, (f) Reactive Reality GmbH, (g) Microsoft, (h) Disney Enterprises Inc., (i) Microsoft, (j) dpa, (l) Museum of Stolen Art, (m) NoReal.it and (n) CIMMI Quebec, respectively.

already impact the global lighting condition within the scene. Being able to precisely predict the result would open up new possibilities.

The same applies to other design processes, for example in the automotive sector, where building real prototypes is expensive and time consuming. Virtual prototyping in real-world conditions would improve the decision process (see Figure 1.3d and 1.3e).

Similar to the furnishing example, nearly every kind of home shopping experience could be enhanced by having coherent AR product presentations, like with a sophisticated AR wardrobe (see Figure 1.3f). Furthermore, the research is relevant for AR games, as users expect more and more enhanced graphics (see Figure 1.3g). Coherent AR can also be used in movie productions or in the process of creating a theater play by providing previews of the final scene and the created mood to the director at early stages (see Figure 1.3h) – besides the obvious marketing purpose (see Figure 1.3i).

Another important area of application is cultural heritage and the usage in museums and exhibitions. Bringing back virtual versions of lost or destroyed artifacts would open up new possibilities. Visitors as well as professional historians can see and discuss about how certain objects must have been looked like centuries ago before they got damaged or aged over time (see Figure 1.3j to 1.3n). In addition, conferred exhibits can still be displayed using virtual replicas in form of interactive 3D objects, that can be observed from all angles, rather than showing showing simple placeholder images. Of course, it is also possible to take virtual exhibits out of the museum to investigate them in an arbitrary light condition.

1.2 PROBLEM STATEMENT

With ARCore[1] and ARKit[2], augmented reality is getting more attention these days. Nevertheless, (photorealistic) AR has been a topic of research for more than 30 years now [Nak+86] and various sophisticated methods have been presented that address coherent rendering. Plausible real-time augmentations of live camera streams that blend seamlessly however, are not yet satisfactorily resolved, especially for mobile devices. While several methods, reviewed in Chapter 4, address partial aspects of that great goal, the state-of-the-art is still far from being able to augment arbitrary environments in an interactive, plausible and coherent fashion. To get closer to that goal and to enable the presented applications, the following problems have to be addressed:

PROBLEM 1: ENVIRONMENT ACQUISITION As this is the main differences to VR and the classical rendering for games and visualization purposes, the acquisition, recognition and understanding of the real-world environment is of particular importance for AR. In the context of visual coherence, the problem consists of the reconstruction of the geometry and materials of real-world objects and in determining the current light condition. Assuming that individual components do not change over time eases the problem. However, the eventual goal is to enable augmentations in fully dynamic environments. Specifically, the high dynamic range of radiance present in real environments and the complex view-dependent material properties of real surfaces are the main challenge while addressing this problem.

1 Google. *ARCore*. Project Website, 2017.
2 Apple. *ARKit*. Project Website, 2017.

PROBLEM 2: ILLUMINATION OF VIRTUAL OBJECTS BY THE ENVIRONMENT Given a complete reconstruction of the real environment, the illumination of virtual objects can be computed by classical light transport simulations. In case of an incomplete or estimated reconstruction, the plausible rendering of virtual objects is not yet solved without user specified information. Even when achieving correct light transport results, the perceived appearance of virtual objects may not be optimal due to properties of the sensors that acquired the environment data in the first place or due to the display component that presents the final image. For coherent augmentations, the AR system has to account for these unavoidable limitations.

PROBLEM 3: RELIGHTING OF THE REAL ENVIRONMENT Inserting virtual objects has a global impact on the light transport within the scene. Light paths are blocked or redirected when hitting the new element. This results in mutual light interactions between virtual and real objects, which are notable as shadows and reflections of direct light from the sources, but also from light that bounced one or multiple times of the virtual or real surface. To determine these effects correctly requires global light transport simulations that rely heavily on a proper scene reconstruction. Given a sophisticated representation of the scene, there is still an immense computational effort – a problem shared by all areas that aim for photorealistic rendering.

PROBLEM 4: INTERACTIVITY AND MOBILE DEVICES Solving *Problem 1* to *3* is not straightforward. Additionally, we are interested in providing these coherent augmentations at interactive rates, so that a user can explore and manipulate the virtual elements. To ensure this necessary property [Azu97], especially on mobile hardware, we have to concentrate on the most perceptible properties first and thereby maximize the visual coherence with the limited computational power. The essence of this problem is well known in many areas, not just computer graphics: finding a compromise between quality, performance and resource consumption.

1.3 SUMMARY OF CONTRIBUTIONS

The methods developed and presented in this thesis address the aforementioned problems and thereby form a step towards the goal of reaching interactive coherent augmented reality. This dissertation explores new methods for acquiring the illumination condition in the current scene and for making that information available to AR rendering pipelines. While focusing on visually coherent results, we concentrate on physically-based simulations that provide a common interface for future works, that build upon our methods and many other techniques in the field of physically-based rendering. The developed algorithms and systems are designed for the usage in interactive systems, but also for scalability to have a long term impact and relevance for future, more powerful and capable hardware generations.

DISTRIBUTED NEAR-FIELD ILLUMINATION In this work, we present a, distributed illumination approach for consistent illumination of virtual objects with direct light, indirect light (color bleeding) and shadows of primary and strong secondary sources. Due to constraints of the mobile device, we split the computation into two parts: First, the real-world radiance is captured by a number of High Dynamical Range (HDR) video cameras that are placed at

different locations in the scene such that each part of the scene is visible to at least one camera (addressing *Problem 1*). The acquired image data is evaluated on a stationary PC and parameters for an illumination model are transferred to the mobile devices. Here, in the second step, the consistent illumination for the virtual objects is computed and displayed to the user at interactive frame rates (*Problem 2* and *4*). This also involves shadows cast from virtual onto real objects and vice versa, which is part of *Problem 3*. In summary, our main contributions are:

- a new distributed approach for interactive photorealistic AR under dynamic real-world environment lighting
- and a lighting model for correct near-field illumination with compact parametrization to be transferred to one or multiple display devices.

TILED FRUSTUM CULLING In this part, we mainly focus on *Problem 4* and show how to reduce the computational cost per reconstructed light using a combination of tile-based rendering and frustum culling techniques. Assuming a lighting model based on discrete sources, like the one in our distributed illumination technique, we identified the possible region of direct influence by such a light as the shadow volume defined by the light and the virtual object. Exploiting this observation results in a considerable performance gain, in comparison to the previous technique. The suggested data-structure, which provides information about relevant light sources for particular regions on the screen, also encourages the use of a more recent tile-based forward rendering, which to our knowledge is being applied to AR for the first time. This, in turn, allows the display of translucent virtual objects, which has not been supported by our system so far and thereby targets *Problem 2*, too. In summary, our main contributions are:

- two culling strategies to speed up the previous system without introducing an additional bias
- and tiled forward shading for efficient AR rendering with support for transparency, correct alpha blending and multisampling.

NATURAL ENVIRONMENT ILLUMINATION The previous methods required a more or less complex hardware setup, including a tracking system, additional cameras and a stationary PC. We also required a manual reconstruction of the scene geometry and materials. Since then, our goal was to avoid this hardware setup to increase the acceptance and thereby to expand the possible field of application. In this last part, a system of techniques is presented, which allows to acquire the environment and to apply the gathered data for coherent rendering using a single mobile device. While the Monte Carlo-based rendering (see Section 2.6.2) aims for high quality renderings that consider a global light transport, we suggested a scalable approach that can produce convincing augmentations while maintaining a barely interactive experience on the mobile device. This part of the thesis thereby concentrates mainly on the *Problem 1* to *3*. By estimating the unknown internal processes of the mobile camera sensor, we also provide a flexible tool for color adjustment between different devices. This allows coherence in terms of camera simulation [SH16] for various scenarios that involve different hardware, e.g., a professional scanning device and arbitrary display devices, which would be especially useful for the mentioned cultural heritage applications. In summary, our main contributions are:

- an estimation of the unknown image optimizations performed by mobile camera sensors,
- the usage of this estimation to acquire the HDR radiance of an entire scene consistently with respect to a reference frame,
- a Graphics Processing Unit (GPU)-optimized surface normal estimation by local least squares plane fitting,
- the introduction of impostor tracing [Szi+05] for photorealistic AR rendering, combined with GPU importance sampling [CK07],
- the usage of a new sampling strategy tailored for AR, which makes use of the observations in *tiled frustum culling*
- and using the inverse operation of the estimated camera optimizations to reduce notable differences between virtual and real objects in terms of perceived color and brightness.

1.4 PUBLICATIONS

Most parts of this dissertation are already published in peer-reviewed conference proceedings or as journal articles. The four papers in which I am second author, are only loosely related to AR. The rest of the following publications are directly relevant for this thesis and are incorporated in Chapters 5 to 7. Although in these sections "we" is usually meant as "the authors", I contributed the presented methods, unless otherwise indicated.

[RJG17] K. Rohmer, J. Jendersie, and T. Grosch. "Natural Environment Illumination: Coherent Interactive Augmented Reality for Mobile and non-Mobile Devices." In: *IEEE Transactions on Visualization and Computer Graphics (Proc. ISMAR)* 23.11 (2017), pp. 2474–2484.

[Jen+17] J. Jendersie, K. Rohmer, F. Brüll, and T. Grosch. "Pixel Cache Light Tracing." In: *Proc. Vision, Modeling and Visualization (VMV)*. 2017, pp. 137–144.

[Gün+16] T. Günther, K. Rohmer, C. Rössl, T. Grosch, and H. Theisel. "Stylized Caustics: Progressive Rendering of Animated Caustics." In: *Computer Graphics Forum (Proc. Eurographics)* 35.2 (2016), pp. 243–252.

[RG15] K. Rohmer and T. Grosch. "Tiled Frustum Culling for Differential Rendering on Mobile Devices." In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2015, pp. 37–42.

[Roh+15] K. Rohmer, W. Büschel, R. Dachselt, and T. Grosch. "Interactive Near-Field Illumination for Photorealistic Augmented Reality with Varying Materials on Mobile Devices." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 21.12 (2015), pp. 1349–1362.

[GRG14] T. Günther, K. Rohmer, and T. Grosch. "Particle-based Simulation of Material Aging." In: *GPU Pro 5: Advanced Rendering Techniques*. Ed. by W. Engel. A K Peters / CRC Press, 2014. Chap. 3, pp. 35–53.

[Roh+14] K. Rohmer, W. Büschel, R. Dachselt, and T. Grosch. "Interactive Near-Field Illumination for Photorealistic Augmented Reality on Mobile Devices." In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2014, pp. 29–38.

[GRG12] T. Günther, K. Rohmer, and T. Grosch. "GPU-accelerated Interactive Material Aging." In: *Proc. Vision, Modeling and Visualization (VMV)*. 2012, pp. 63–70.

1.5 OUTLINE

The remaining parts of this thesis are organized as follows: Chapter 2 introduces fundamental basics of light transport and related work on classical light transport algorithms as well as real-time techniques. Chapter 3 continues with fundamentals in mixed and augmented reality. Chapter 4 provides an overview on techniques used for coherent AR. This includes approaches for reconstructing the local scene, i.e., geometry, materials and light, a section on differential rendering, and a review of related work on interactive coherent rendering. Chapter 5, 6 and 7 form the core of this thesis and cover the developed techniques of [Roh+14, Roh+15], [RG15] and [RJG17], respectively. Each of them is explained in detail and evaluated independently. Chapter 8 concludes this dissertation with a summary and a section on yet unsolved problems in coherent AR rendering.

2

FUNDAMENTALS IN LIGHT TRANSPORT

2.1 RADIOMETRIC QUANTITIES

In this chapter, the most important radiometric quantities for rendering are introduced, based on the deliberations of ERIC VEACH [Vea97]. Throughout the presented dissertation, as in most other works in the field of computer graphics, light is described by *geometric optics*, i.e., the particle theory. Even though this covers most of what we see in our everyday life, some effects like *diffraction* or *fluorescence* can only be explained by also considering *wave optics* or *quantum optics*, respectively. Note, that the notation used in this thesis is simplified and thereby informal. One issue is that light quantities are treated as continuous functions, although photons are actually discrete elements. Since we are dealing with large numbers and photons in rendering, this is not a problem in practice [PJH16]. For a more formal and detailed development of the here presented quantities as well as more complete explanation of light models and light transportation theory, I refer the reader to the very well written dissertation of ERIC VEACH [Vea97]. Before starting with the quantities the basic concept of the solid angles and the notation for spherical integrals are explained. The second part of this chapter gives an overview of fundamental light transport simulations and real-time approximations that are relevant for the related work in AR as well as for the developed techniques presented in the main part of the thesis.

SOLID ANGLE An angle α in the 2D plane can be described by two rays sharing the same origin. It is measured by the quotient s/r, where s is the length of a spherical arc around that origin at radius r. As the circumference of a circle with radius r equals $2\pi r$, the angle of a full circle is 2π measured in radians [rad], a dimensionless SI derived unit. Generalizing this idea to 3D leads to the *solid angle* Ω, defined by the quotient in Equation (2.1). Here, A is a part of the surface area of the sphere like s was a part of the circumference of the circle. Since the total surface area of a sphere equals $4\pi r^2$, the solid angle of a full sphere is 4π measured in steradians [sr], also a dimensionless SI derived unit:

$$\Omega = \frac{A}{r^2}. \tag{2.1}$$

The angle in 2D can also be interpreted as a set or bundle of all directions between the two outer rays specifying the arc. Similarly, the solid angle can be interpreted as set of directions in 3D. When this set shapes a cone, the area on the sphere surface becomes a spherical cap. In general, this regular shape is not required, so the solid angle of an arbitrary shaped area S on the sphere surface can be calculated as a surface integral, where θ and φ are the spherical coordinates of the direction of the infinitesimal surface element ∂s:

$$\Omega = \int_S \frac{\partial s}{r^2} = \int_\varphi \int_\theta \sin\theta \; \partial\theta \; \partial\varphi.$$

Furthermore, the solid angle of an arbitrary oriented object subtended at a point $\mathbf{c}$ equals the solid angle of the projection of that object onto a unit sphere with center at $\mathbf{c}$. The solid angle can then be computed by the same surface integral.

In the following, a solid angle into direction (θ, φ) will be denoted as $\boldsymbol{\omega}$. Consequently, for an integral over a sphere, originating at position $\mathbf{x}$ on a surface, a simplified notation will be used:

Integration over the Spherical Shell

$$\int_{\mathcal{H}_\pm} \partial\boldsymbol{\omega} = \int_0^{2\pi} \int_0^{\pi} \sin\theta \; \partial\theta \; \partial\varphi,$$

where θ measures the angle to the surface normal $\boldsymbol{N}(\mathbf{x})$ at point $\mathbf{x}$. The basis for the measurement of φ is arbitrary in rotation invariant cases or depends on other surface properties like tangent directions.

The tangent plane at the surface point $\mathbf{x}$ divides the sphere into two hemispheres, namely the *upward hemisphere*, indicated by a $\mathcal{H}_+$ as subscript and the *downward hemisphere*, using a $\mathcal{H}_-$. Hence, $\mathcal{H}_\pm$ is used for integrals over the entire sphere.

RADIANT FLUX It is defined as the total amount of radiant energy Q per second. Flux also describes the total amount of energy passing through a surface per time. The radiant flux, denoted by Φ, is measured in joule per second [J/s] or watt [W]:

$$\Phi = \frac{\partial Q}{\partial t}.$$

Definition of Radiant Flux

This implies that the radiant energy Q is also a function over time, measuring the energy of photons emitted by a surface over a time interval.

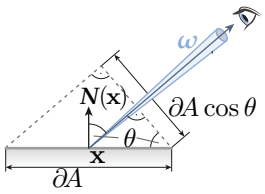


Figure 2.1: Projected Area
The visible area of the surface element at $\mathbf{x}$ decreases with the cosine of angle θ between the surface normal $\boldsymbol{N}(\mathbf{x})$ and the direction of the viewing angle $\boldsymbol{\omega}$. This is also known as *Lambert's cosine law*.

IRRADIANCE AND RADIANT EXITANCE Defined as radiant flux per unit surface area, the *irradiance* is denoted by E and measured in watt per square meter [W/m²]:

$$E(\mathbf{x}) = \frac{\partial \Phi}{\partial A(\mathbf{x})}. \tag{2.2}$$

Definition of Irradiance

Even though the parameter $\mathbf{x}$ is often omitted, the quantity is always defined with respect to a point $\mathbf{x}$ on the surface. The term irradiance also implies the measurement of incident energy, which is often restricted to the upward hemisphere specified by the surface normal $\boldsymbol{N}(\mathbf{x})$. However, through emission and scattering, both hemispheres can be involved and the restriction depends on material model used.

When energy is leaving the surface, the term *radiosity* denoted by the symbol B or *radiant exitance* M is used. The latter is preferred, because it avoids confusion with the light simulation technique (see Section 2.6.7).

RADIANT INTENSITY Since the energy emitted by light sources is not distributed evenly over all possible directions, this quantity is used to describe the emitted flux depending on the angle of observation θ, which is usually measured in some local coordinate frame of the sender, e.g., the angle to surface normal in case of area light sources. More general, the *Intensity* measures radiant flux emitted, received, reflected or transmitted per solid angle and its unit is watt per steradian [W/sr]:

$$I(\theta) = \frac{\partial \Phi}{\partial \boldsymbol{\omega}}. \tag{2.3}$$

Definition of Intensity

A hypothetical isotropic point light source, used in many applications because of their simplicity, emits flux equally in all directions. Thereby, its intensity is a constant function:

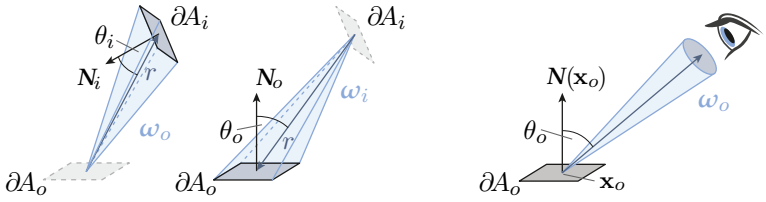
$$I(\theta) = \frac{\Phi}{4\pi}.$$

RADIANCE Perhaps the most frequently used and the most important quantity for light transport is the view-dependent *radiance*, defined as the flux per projected area per unit solid angle:

$$L(\mathbf{x}, \boldsymbol{\omega}) = \frac{\partial^2 \Phi}{\partial A(\mathbf{x}) \cos\theta \; \partial \boldsymbol{\omega}}. \tag{2.4}$$

Definition of Radiance

Measured in [W/m² sr], radiance can also be defined as intensity per projected area. To compute the radiance of a surface point $\mathbf{x}$ observed from a small solid angle $\boldsymbol{\omega}$, we could count the number of photons per unit time (the flux) passing through a small hypothetical surface perpendicular to the direction



(a) Solid Angles (b) Radiance

Figure 2.2: Solid Angles and Radiance
Visualization of a solid angles originating at the sender and the receiver (a). Applying this notation to the definition of radiance (b).

of $\boldsymbol{\omega}$. Projecting the real surface area at position $\mathbf{x}$, we are interested in, onto that small hypothetical surface, leads to a smaller area to be measured. The exact decrease in area depends on the angle of observation, which is accounted for by the cosine term (see Figure 2.1). In summary, the radiance emitted or received by an infinitesimal surface area depends on the orientation of that surface, defined by the surface normal $\boldsymbol{N}(\mathbf{x})$ and the viewing angle, where θ measures the angle between the normal and the direction of $\boldsymbol{\omega}$.

There is a fundamental difference between *incident* and *exitant* radiance, observed at the surface. The first describes a "set of photons" – actually flux – just before arriving at the surface, whereas the second specifies a "set of photons" right after leaving. The relationship between them – defined by the Bidirectional Scattering Distribution Function (BSDF) (see Section 2.4) – can be very complex depending on the model used to describe the material. Incident and exitant radiance are distinguished according to the interpretation of the parameter $\boldsymbol{\omega}$. $L_i(\mathbf{x}_i, \boldsymbol{\omega}_i)$ is used to describe radiance visible at surface point $\mathbf{x}_i$ from direction $\boldsymbol{\omega}_i$, whereas $L_o(\mathbf{x}_o, \boldsymbol{\omega}_o)$ denotes outgoing radiance, the radiance at surface point $\mathbf{x}_o$ observable from direction $\boldsymbol{\omega}_o$.

2.2 Computing Light Transport

It is useful to define the solid angle subtended by an arbitrarily oriented (small) sending or receiving surface element as illustrated in Figure 2.2a:

$$\partial\omega_o = \frac{\partial A_i \cos\theta_i}{r^2} \quad \text{and} \quad \partial\omega_i = \frac{\partial A_o \cos\theta_o}{r^2}, \tag{2.5}$$

Definition of Solid Angles

where θ is again the angle between the direction of $\boldsymbol{\omega}$ and the surface normal. The distance between the origin of $\boldsymbol{\omega}$ and the center of the opposing surface element is again denoted as r. The subscripts used for both definitions indicated whether the cone of the solid angle starts at the center of the sender patch and spans to include the receiver or vice versa. An $(\cdot)_o$ is used for all quantities measured at the surface with outgoing light including the solid angle that originates at the sender. For quantities measured at the receiving surface, where light is incident, the subscript $(\cdot)_i$ is used. Applying this notation to the definition of the radiance in Equation (2.4) leads to:

$$L_o(\mathbf{x}_o, \boldsymbol{\omega}_o) = \frac{\partial^2 \Phi_{o\to i}}{\partial A_o \cos\theta_o \, \partial\boldsymbol{\omega}_o}, \tag{2.6}$$

where a small sender surface located at $\mathbf{x}_o$, observed from an outgoing solid angle $\boldsymbol{\omega}_o$, is considered (see Figure 2.2b).

Changing the perspective allows to compute irradiance at the receiver instead. Therefore, we define an extended function that describes the radiance based on the center positions of the sender and the receiver elements, with distance $r = \|\mathbf{x}_i - \mathbf{x}_o\|_2$, as well as the solid angle based on Equation (2.6):

$$\begin{aligned} L_{o\to i}(\mathbf{x}_o, \mathbf{x}_i, \boldsymbol{\omega}_o) &= \frac{\partial^2 \Phi_{o\to i}}{\partial A_o \cos\theta_o \; \partial \boldsymbol{\omega}_o} \\ &= \frac{\partial^2 \Phi_{o\to i} \; r^2}{\partial A_o \cos\theta_o \; \partial A_i \cos\theta_i} \qquad (2.7) \\ \Rightarrow \qquad L_{o\to i}(\mathbf{x}_o, \mathbf{x}_i, \boldsymbol{\omega}_i) &= \frac{\partial^2 \Phi_{o\to i}}{\partial A_i \cos\theta_i \; \partial \boldsymbol{\omega}_i}. \qquad (2.8) \end{aligned}$$

During this transformation, Equations (2.5) have been used to switch the solid angle from the sender to the receiver. The notations, typically used for radiance, can be defined by the following simplifications. Note, that radiance is generally dependent on the sender and receiver elements. Yet one of the parameter is omitted, as the solid angle encodes redundant information. The receiver position $\mathbf{x}_i$ is omitted for the function of outgoing radiance $L_o(\mathbf{x}_o, \boldsymbol{\omega}_o)$ and vice versa for the incoming radiance. The subscripts at L and x are usually also neglected to get an even shorter notation, since the index of the solid angle is sufficient for a unique definition. Note, however, that the positions $\mathbf{x}_i$ and $\mathbf{x}_o$ in the equations above are different, and so are the $\mathbf{x}$ in the shortest notation:

$$\begin{aligned} L(\mathbf{x}, \boldsymbol{\omega}_o) &:= L_o(\mathbf{x}_o, \boldsymbol{\omega}_o) := L_{o\to i}(\mathbf{x}_o, \mathbf{x}_i, \boldsymbol{\omega}_o) \\ L(\mathbf{x}, \boldsymbol{\omega}_i) &:= L_i(\mathbf{x}_i, \boldsymbol{\omega}_i) := L_{o\to i}(\mathbf{x}_o, \mathbf{x}_i, \boldsymbol{\omega}_i)\,. \end{aligned}$$

From Equation (2.7), we can derive the *Basic Law of Radiometry and Photometry*, which emphasizes that the flux transmitted from the sender to the receiver is proportional to the radiance. The proportionality constant is defined by the orientation, size and distance of the corresponding surface elements. This means, that if the emitted flux of a sender is doubled, the surface will appear twice as bright when observed from the receiver position:

Basic Law of Radiometry and Photometry

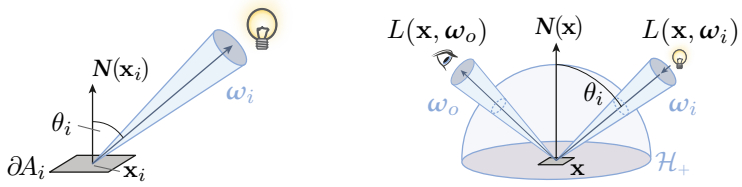
$$\partial^2 \Phi_{o\to i} = L(\mathbf{x}, \boldsymbol{\omega}_i) \; \frac{\partial A_o \cos\theta_o \; \partial A_i \cos\theta_i}{r^2}, \qquad (2.9)$$

However, the more interesting relationship can be derived by inserting the definition of the irradiance, Equation (2.2), into Equation (2.8):

$$\begin{aligned} L_i(\mathbf{x}_i, \boldsymbol{\omega}_i) &= \frac{\partial E(\mathbf{x}_i)}{\cos\theta_i \; \partial \boldsymbol{\omega}_i} \\ \Rightarrow \qquad \partial E(\mathbf{x}_i) &= L_i(\mathbf{x}_i, \boldsymbol{\omega}_i) \cos\theta_i \; \partial \boldsymbol{\omega}_i. \qquad (2.10) \end{aligned}$$

which eventually allows to compute the irradiance from incoming light of an infinitesimal solid angle $\partial \boldsymbol{\omega}_i$ at a receiver position $\mathbf{x}_i$. Analogous to the visualization of the radiance in Figure 2.2b, the irradiance computation is illustrated in Figure 2.3a. Integrating Equation (2.10) over the upward hemisphere allows to compute the irradiance at the position $\mathbf{x}_i$ based on the total incident radiance (incoming from the upward hemisphere):

$$E(\mathbf{x}_i) = \int_{\mathcal{H}_+} L_i(\mathbf{x}_i, \boldsymbol{\omega}_i) \cos\theta_i \; \partial \boldsymbol{\omega}_i. \qquad (2.11)$$



(a) Irradiance (b) Rendering Equation

Figure 2.3: Irradiance and the Rendering Equation

Quantities and their notation for computing the irradiance at surface element $\mathbf{x}$ (a) and for the evaluation of the rendering equation (b). Note, that the naming convention for solid angles in (b) is coherent. First, we consider $\mathbf{x}$ to be receiver of incident light from $\boldsymbol{\omega}_i$ in accordance to (a). Then, we consider $\mathbf{x}$ as sender of light towards to observer solid angle $\boldsymbol{\omega}_o$, as visualized earlier in Figure 2.2b. Hence, we refer to the surface element just as $\mathbf{x}$, when considering incident and exitant radiance.

The resulting integral of Equation (2.11) is the foundation of the most important formula for physically-based rendering and light transport simulations, the *Rendering Equation* (2.12) of Kajiya, which models the interaction between light and matter in a general way [Kaj86]:

Kajiya's Rendering Equation

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = L_e(\mathbf{x}, \boldsymbol{\omega}_o) + \int_{\mathcal{H}_+} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, L(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i \; \partial\boldsymbol{\omega}_i. \tag{2.12}$$

The outgoing radiance $L(\mathbf{x}, \boldsymbol{\omega}_o)$ at position $\mathbf{x}$ on the surface, e.g., towards an observer in direction $\boldsymbol{\omega}_o$, is composed of the self emission, $L_e(\mathbf{x}, \boldsymbol{\omega}_o)$, and the accumulated reflected incident radiance from all directions $\boldsymbol{\omega}_i$ in the upward hemisphere above $\mathbf{x}$. Compared to Equation (2.11) the irradiance of a single solid angle is weighted by the material response, the Bidirectional Reflectance Distribution Function (BRDF) $f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$, which controls how much of the incident light is reflected into a certain outgoing direction and thereby is responsible for modeling the appearance of the material at position $\mathbf{x}$. The BRDF and the more general BSDF will be discussed in Section 2.4. Figure 2.3b shows an illustration of the sphere integral and the corresponding quantities.

Note, that the definition of the outgoing radiance at point $\mathbf{x}$ depends on the radiance of any visible point in the upward hemisphere. These in turn also depend on all points in their upward hemisphere, including $\mathbf{x}$. Hence, there is a recursion that has to be evaluated to an infinite depth at which it describes the equilibrium of light in the scene. Since this is not feasible to compute, the problem is reformulated, simplified or computed to a certain depth of recursion as described in Section 2.6.

Before getting to the BSDF and to the *Lambertian emitter*, the last part of the section on fundamental terms and definitions, connections to the representation of light in photometry and colorimetry are drawn.

2.3 Spectral Radiance, Photometry and Colorimetry

As stated earlier, we use geometric optics to describe light in compute graphics. Nevertheless, it is possible to consider different wavelengths when simulating light. This leads to the definition of spectral quantities and one more variable

to integrate during the light transfer computations. The spectral radiance for instance is defined as follows:

$$L_\lambda(\mathbf{x}, \boldsymbol{\omega}, \lambda) = \frac{\partial^3 \Phi_\lambda}{\partial A(\mathbf{x}) \cos\theta \; \partial\boldsymbol{\omega} \; \partial\lambda},$$

where L_λ is the derivative of the radiance with respect to wavelength, $L_\lambda = \frac{\partial L}{\partial \lambda}$. Even though the wavelength is not considered in most computer graphics applications, it can have strong visual impact. For instance when incident white light is scattered at a dispersive prism resulting in a spectrum of refracted colors because of a wavelength-dependent refractive index. This is important for rendering objects made of glass or gems like jewelery. For the definitions above, we integrated over all wavelengths to deal with the total radiation:

$$L = \int_0^\infty L_\lambda \; \partial\lambda.$$

Most of the applications in computer graphics descretize the spectrum in three channels: red, green and blue respectively. Note, that the computation is not only simplified by using three instead of an infinite number of wavelengths but the channels are also treated independent of each other. This means there is no energy transport between the different color channels. However, the way the spectrum is discretized is well defined in the connected fields of *photometry* and *colorimetry*.

While radiometry is the study of the propagation of electromagnetic radiation dealing with radiation of all wavelengths, in photometry the light as perceived by the human eye is in focus. So the interval of wavelengths between 380 nm and 830 nm is of particular interest. The human eye is not sensitive to wavelengths outside of this range and the sensitivity varies within the interval. Additionally, the perceived brightness for different spectral components differs slightly between individuals. Assuming that the sensitivity to all wavelengths can be modeled by a single curve that works for any human, the *Commission Internationale de l'Eclairage* (CIE) proposed the $V(\lambda)$-*curve* [Int96]. This curve can be used as weighting function to convert between radiometric and photometric quantities [Rei+10]. As an example, the *luminance* which is the photometric equivalent of *radiance*, can by obtained by:

$$L_v = 683\,\mathrm{lm/W} \int_{380\,\mathrm{nm}}^{830\,\mathrm{nm}} L_\lambda V(\lambda) \; \partial\lambda.$$

The base unit of photometric quantities is lumen [lm] instead of watts used for the radiometric counterparts. The unit of luminous intensity is called candela [cd], which is defined as lumen per steradian [lm/sr].

The luminance, as defined above, measures the brightness perceived by humans. Since we are interested in colored rather than monochromatic images and because of a varying sensitivity for different wavelengths, experiments have been conducted in the field of colorimetry to understand and model the human color vision. Eventually, the CIE defined the $\bar{x}\bar{y}\bar{z}$-color-matching functions[3] that can be used for weighting radiometric quantities, just like the $V(\lambda)$-curve, to finally obtain CIE-XYZ color triplets which are the foundation for other color spaces used in practice. The RGB-colors and sRGB-colors for instance, which represent a subspace of the XYZ domain, can be obtained by

3 CIE, *1931 Standard Colorimetric Observer*. ISO Standard, 2017.

applying standardized transformations. For sRGB this can be a simple linear matrix defined by the *International Telecommunication Union* (ITU)[4]. For more details I refer to HDRI book of Reinhard *et al.* [Rei+10].

In this thesis the radiometric terms are used as it is common in computer graphics literature even though we are interested in images perceived by humans and most of the computations are conducted in RGB space. Therefore, the wavelength is ignored and quantities are interpreted as scalar functions for each color channel.

2.4 Bidirectional Scattering Distribution Functions

When a photon hits the surface it is either reflected, refracted or absorbed. The properties of the material define the probability of the individual events. In case of absorption, light is transformed into another form of energy, such as heat[5]. Since we are only interested in visible light, the absorbed energy is usually not regarded in computer graphics. The remaining light energy is scattered, i.e., reflected back into environment or refracted through the surface into the object.

When all scattering events happen at the hitpoint, the scattering can be described by a local model. In practice, "at the hitpoint" means that light, which is entering matter, will leave within the sampling area, i.e., within the same pixel, resulting in a relaxation of the constraint. A BSDF is such a local model, defined as multi-dimensional function that models this scattering behavior. It can be expressed as ratio between outgoing reflected radiance and irradiance at surface position $\mathbf{x}$:

$$f(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{\partial L(\mathbf{x}, \boldsymbol{\omega}_o)}{\partial E(\mathbf{x}, \boldsymbol{\omega}_i)}. \tag{2.13}$$

When light is not refracted and thereby is either absorbed or scattered back into the incident hemisphere, the behavior can be described by a simpler class of models, the BRDFs denoted as f_r. This is obviously not true for translucent materials like glass or fluids such as water. To model the behavior of refracted light another class of function, the Bidirectional Transmittance Distribution Functions (BTDFs) denoted as f_t, are used. To draw a connection between these models, a BSDF can be considered as a union of two BRDFs, one for each hemisphere, and two corresponding BTDF; or simply as a practical generalization where BRDF and BTDF are special cases [Vea97].

Some semi-translucent materials like marble, wax, thin leaves or human skin are prominent examples that show multiple scattering events beneath the surface before the light is exiting at a different position. Hence, they cannot be described properly by local models. To model these volumetric effects, further material properties need to be defined, which can become very complex in case of inhomogeneous media. The more general Bidirectional Surface Scattering Reflectance Distribution Functions (BSSRDFs) can describe non-local scattering behavior for the rendering materials, such as marble, skin, milk, etc. Compared to Equation (2.13), a BSSRDF therefore depends on different locations for incident and outgoing light [Jen+01]:

$$S(\mathbf{x}_i, \boldsymbol{\omega}_i, \mathbf{x}_o, \boldsymbol{\omega}_o)\,.$$

4 ITU. *ITU Recommendation BT.709*, 2017.

5 Absorbed light can also be re-emitted. In computer graphics, this effect is modeled as part of the diffuse scattering.

However, for this thesis, we are mainly concerned about opaque objects and therefore apply a set of simple BRDFs. These models have some known properties that are important for physically-based rendering. A BRDF that fulfills all three of them is considered to be physically plausible [Lew94]. The corresponding properties in the context of BSDFs are discussed in the thesis of Eric Veach [Vea97].

POSITIVITY The value of a BRDF is not negative. Since the value is related to the probability of reflecting incident light into a certain direction, the value itself has to be positive or zero:

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \geq 0. \tag{2.14}$$

Positivity of BRDFs

ENERGY CONSERVATION The total amount of energy reflected by a surface must be less or equal to the incident irradiance. A non-emissive material cannot emit more light than received:

$$\int_{\mathcal{H}_+} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \cos\theta_i \; \partial\boldsymbol{\omega}_i \leq 1 \qquad \forall\, \boldsymbol{\omega}_o \in \mathcal{H}_+. \tag{2.15}$$

Energy Conservation of BRDFs

HELMHOLTZ RECIPROCITY BRDFs are symmetric. Hence, changing incident and exitant direction does not change the amount of reflection:

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = f_r(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i) \qquad \forall\, \mathbf{x}, \forall\, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o \in \mathcal{H}_+. \tag{2.16}$$

Symmetry of BRDFs

Based on these properties we can now define some basic physically plausible BRDFs that will be used throughout this thesis: One of the easiest reflectance models is the *Lambertian* reflectance, which assumes that incident light is scattered equally to all directions in the upward hemisphere. Hence, the reflectance is independent of the observer direction $\boldsymbol{\omega}_o$ and looks equally bright when seen from arbitrary positions. The model is also referred to as *diffuse material* and is used to render rough dielectrics:

$$f_d(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{\rho_d}{\pi} \qquad \rho_d \leq 1,$$

where the constant reflectance coefficient ρ_d is used to control the color of the modeled material. The factor $1/\pi$ is used for normalization and can be derived by integrating the left-hand side of Equation (2.15). For a constant reflection of $f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = 1$, the result of the definite integral equals π. To ensure the energy conservation, the reflectance is scaled by the reciprocal.

The simplest of all reflectance functions is the perfect reflection. It follows the well-known *Law of Reflection*: $\theta_i = \theta_o$ and thereby describes the material of a perfect (tinted) mirror:

$$f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \rho_s \, \delta(\psi) \qquad \rho_s \leq 1,$$

where δ is the *Dirac delta function* and ψ is the angle between $\boldsymbol{\omega}_o$ and the ideal reflection direction. $\delta(\psi)$ is infinity if ψ is zero and zero otherwise. To control the color of reflections, or to mimic absorption, the specular coefficient ρ_s is used. If components of ρ_s are less than one, we get a tinted mirror that appears to filter light of certain wavelengths.

Such a perfect mirror can be seen as a smooth conductor or as a smooth metal surface. If the metal is rough instead of perfectly smooth, we get the material of a rough conductor showing *glossy specular* reflection. Here, the

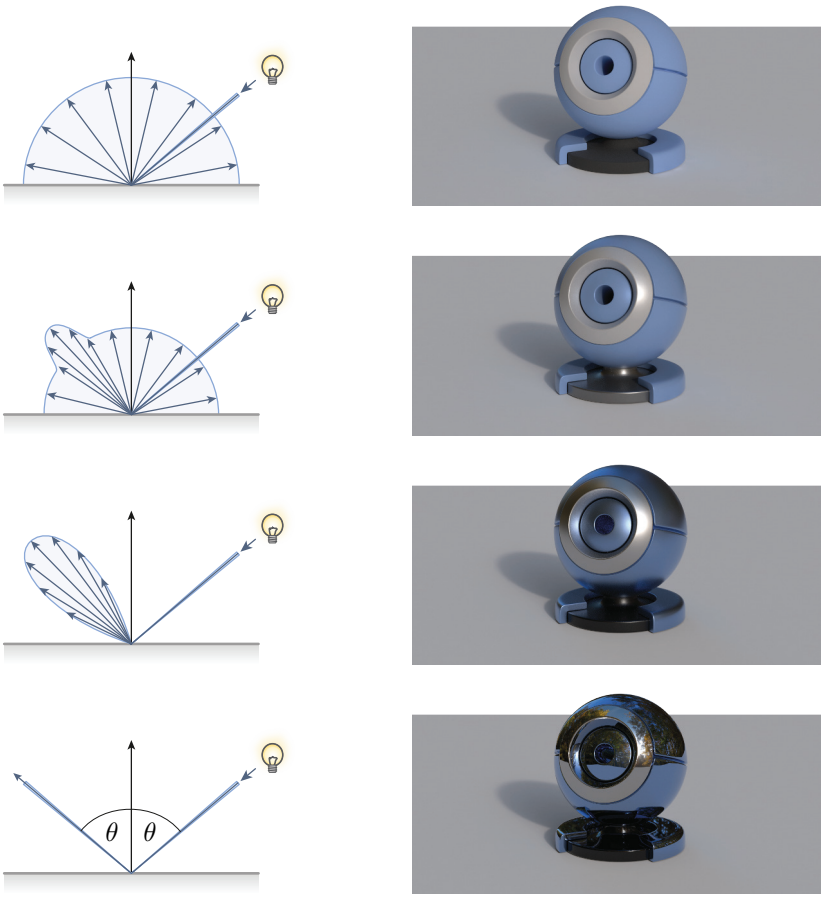


Figure 2.4: Illustration of Common BRDFs
Left: illustrated reflection directions of incident light, while the length of reflected rays correspond to their probability. Right: renderings using the BRDFs to describe the material of the mesh. Top to bottom: rough diffuse, mixed, glossy (rough conductor), smooth conductor. The MATERIAL BALL Mesh is from 3D-COAT and the GOLDEN AUTUMN ROAD Environment Map from HDRMAPS.COM.

reflected ray is likely to be close to the ideal reflection direction but differs depending on the level of roughness. It is common to model this by exponentiation of the cosine function:

$$f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \rho_s \frac{n+1}{2\pi} \cos^n \psi \qquad \rho_s \leq 1,$$

Again, energy conservation is accounted for by adding a normalization term, $(n+1) \,/\, 2\pi$. Note, that n controls the roughness. The higher n the smoother the material and the smaller its highlights. In the limit at $n = \infty$ we also get a perfect mirror.

We are now lacking of a BRDF for more smooth dielectrics. A very common approach to model this kind of materials is to create mixed, or multi-layered materials, which are referred to as *glossy* materials. Therefore, a convex combination of diffuse and specular BRDFs is defined:

$$f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = a_d \; f_d(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) + a_s \; f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \quad a_d + a_s \leq 1, \tag{2.17}$$

where the diffuse and specular amount, a_d and a_s, are specified by the user or computed by the *Fresnel equations* or their computationally cheaper ap-

proximation by SCHLICK [Sch94]. Figure 2.4 shows examples of these BRDFs rendered with image-based lighting in an offline path tracer.

The BRDFs discussed above are very basic. However, they can generally be replaced by more advanced models. For physically-based rendering, most BRDFs are motivated by the *Microfacet Theory* [TS67]. The theoretical framework suggests that a surface of an object is an irregular composition of tiny, reflecting surface elements. These elements are called microfacets and describe the surface structure at microscopic scale. The microfacets form a tiny landscape with self-occlusions and interreflection depending on their variation. In the real world or during rendering we observe these tiny landscapes at a macro scale level where small details are not visible anymore. Since subdivision of the geometry to that level is not feasible, the facets are usually described by statistical models. Details to such BRDFs, containing the Oren-Nayar [ON94] and the Torrance-Sparrow [TS67] model, can be found in the PBRT book [PJH16]. The microfacets theory can also be applied to BSDFs and thereby used "to simulate transmission through rough surfaces such as etched glass" [Wal+07].

Compared to the previously described analytical reflection models, that are usually very specialized for certain types of materials, BSDFs can also be derived from measured data. Famous examples are the BRDFs by WARD [War92], COOK and TORRANCE [CT82] or LAFORTUNE *et al.* [Laf+97], who fitted measured data into analytical models. A more general approach to represent real world materials was presented by MATUSIK *et al.*, who acquired 100 materials that can be used for physically-based rendering [Mat+03]. They are able to reproduce isotropic BRDFs with high degree of realism and offer meaningful parameters to define a certain material appearance.

A more complete representation of measured data can be stored in a Bidirectional Texture Function (BTF) [Dan+99]. This representation allows to reproduce the appearance of an arbitrary surface depending on the spatial position $\mathbf{x}$ as well as on the incident and exitant directions $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$. Therefore, they store the measured data explicitly. Thus, without compression or fitting into analytical models, BTFs are large lookup tables.

An overview of various BRDFs, acquisition methods and their application in physically-based rendering can be found in the recent state of the art report of GUARNERA *et al.* [Gua+16]. More details on scanning real geometry and measuring their reflectance properties was described in the course of WEINMANN *et al.* [Wei+16].

2.5 THE LAMBERTIAN EMITTER

Closely related to the Lambertian BRDF is its foundation, the *Lambertian Emitter*, an idealized area light source with constant radiance, independent of the observer direction $\boldsymbol{\omega}_o$. The properties of this emitter are highly important for the most part of the presented related works as well as the own approaches, not only for describing direct light sources. When a surface with Lambertian reflectance, a perfectly diffuse material, is illuminated, it becomes a Lambertian emitter that radiates indirect light uniformly in all directions of the upward hemisphere. It is therefore responsible for the well-known *color bleeding* effect. Hence, Lambertian emitters are the basis for all diffuse global illumination simulation.

To use Lambertian emitters during simulations, we need to define the properties of such an emitter, beginning with the radiant intensity and its variation depending on the angle to the surface normal. Therefore, we start with the definition of radiance from Equation (2.4) and substitute the definition of the radiant intensity given by Equation (2.3):

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = \frac{\partial I(\theta_o)}{\partial A_o(\mathbf{x}) \cos\theta_o} = L(\mathbf{x})\,,$$

where the outgoing radiance is independent of the outgoing direction of $\boldsymbol{\omega}_o$, according to the definition. The angle between the surface normal at $\mathbf{x}$ and the direction of $\boldsymbol{\omega}_o$ is again denoted as θ_o. By bringing the independent area of the surface element to the left-hand side, we can deduce that intensity has to decrease with the viewing angle too, to eventually result in a constant radiance:

$$L(\mathbf{x})\, \partial A_o(\mathbf{x}) = \frac{\partial I(\theta_o)}{\cos\theta_o} = const.$$

Hence, we can specify the intensity distribution $I(\theta)$, with the help of the maximum intensity $I_\perp$, that is emitted perpendicular to the surface:

$$I(\theta) = I_\perp \cos\theta.$$

Finally, we express radiance as the constant ratio between the maximum intensity and the area of the emitter, which is especially useful in the discretized version:

$$L(\mathbf{x}) = \frac{\partial I_\perp \cos\theta_o}{\partial A(\mathbf{x}) \cos\theta_o} = \frac{\partial I_\perp}{\partial A(\mathbf{x})} \approx \frac{\Delta I_\perp}{\Delta A(\mathbf{x})}. \tag{2.18}$$

The next property we need is the flux. Therefore, we integrate the intensity distribution over the hemisphere. Note, that this accounts for total flux of the emitter:

$$\Phi = \int_{\mathcal{H}_+} I(\theta_o) = \int_{\mathcal{H}_+} I_\perp \cos\theta_o \; \partial\boldsymbol{\omega}_o = \pi I_\perp. \tag{2.19}$$

Eventually, we are interested in the irradiance at a receiver element that gets illuminated by a Lambertian emitter. Therefore, we insert the definition of irradiance, Equation (2.2), into the fundamental law of radiance from Equation (2.9), which gives:

$$\partial E(\mathbf{x}) = L \frac{\partial A_o \cos\theta_o \cos\theta_i}{r^2}.$$

Substituting the radiance and the sender area using the relationship in Equation (2.18) results in irradiance depending on the sender intensity. This also emphasizes, that the irradiance reduces with the distance r to the light source, i.e., there is an inverse-square falloff:

$$\partial E(\mathbf{x}) = \frac{\partial I_\perp \cos\theta_o \cos\theta_i}{r^2}. \tag{2.20}$$

Photometric Distance Law

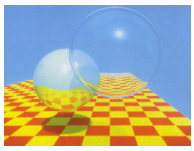
When considering a discretization, e.g., when evaluating the irradiance of a discrete light source for rendering, we can get into problems. For nearby sources with a given non-zero intensity $\partial I_\perp$, there is a singularity because the limit of $\Delta E(\mathbf{x})$, as r approaches zero, is infinity.

2.6 FUNDAMENTAL TECHNIQUES

The following two sections give a brief overview of basic rendering techniques that are used in computer graphics and in the context of AR. Details and more references, especially in terms of interactive variants of the techniques, can be found in the report of RITSCHEL *et al.* [Rit+12].

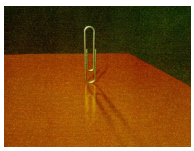
2.6.1 *Ray Tracing*

With its origin in the 1960s, ray tracing is one of the oldest rendering techniques in computer graphics and serves as basis for many other algorithms. In 1980, TURNER WHITTED introduced recursive ray tracing for rendering of specular surfaces while utilizing global illumination information [Whi80]. In the real world, light is emitted from a source, it travels through the scene and encounters multiple reflection or refraction events and eventually hits the camera sensor or the retina of the eye. Ray tracing simulates this in a backward manner. Therefore, rays are generated from the camera position through all pixels of the image plane. They are *cast* into the scene by computing the first intersection point with the geometry. At this *hitpoint*, the direct illumination is evaluated which involves the BRDF and the surface normal at the hitpoint, the light sources in the scene and shadow tests – which are also ray casts – to determine the visibility of sources. Then, reflected and refracted rays are computed based on the *Law of Reflection* and *Snell's Law*. Starting from the hitpoint, they are *traced* recursively through the scene until a certain limit is reached. This limit can be a maximum level of recursion or the hit of a diffuse material at which a Whitted-style ray tracer stops.



Courtesy of WHITTED [Whi80]

A very common optimization is to accelerate intersection tests by using spatial data structures like Bounding Volume Hierarchys (BVHs), Octrees, KdTrees, etc. The complexity of an intersection test in such an acceleration data structure is $\mathcal{O}(\log n)$ on average, compared to $\mathcal{O}(n)$ of a naive search. Another important aspect is that traced rays are computationally independence and thereby can be evaluated in parallel even on a GPU.



Courtesy of COOK *et al.* [CPC84]

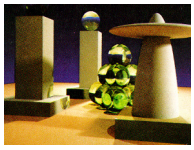
This basic technique can be used for rendering highly specular scenes and provides elegant tool for visibility computation. However, Whitted-style ray tracing does not solve the rendering equation (2.12) as diffuse reflection for instance is not considered at all.

To deal with aliasing and to simulate new effects including glossy specular reflections, depth of field, penumbras and motionblur, COOK *et al.* already used multiple rays per pixel and Monte Carlo sampling [CPC84, Coo86]. This can be seen as a step towards path tracing.

Courtesy of COOK *et al.* [Coo86]

2.6.2 *Path Tracing and Monte Carlo Sampling*

In *Distributed Ray Tracing* by COOK *et al.*, multiple rays sample only a single event of the path, e.g., at the first bounce for glossy reflections [CPC84]. It would result an extreme large number of rays to trace if one applies the sampling for all events.



Courtesy of KAJIYA [Kaj86]

KAJIYA reformulated the problem to Path Tracing. Here, instead of branching into multiple sub-paths, only one of them is selected randomly and traced until a certain recursion depth is reached. By repeating this tracing of random paths for a larger number of rays per pixel, the average of the individual rays

yields the correct radiance for that pixel. He presented the approach along with the rendering equation (2.12) and thereby provided a numerical solution [Kaj86], too.

In contrast to ray tracing, the reflectance of any type of material can be evaluated in a Monto Carlo fashion. This means that the integral of Equation (2.12), that needs to be solved for each bounce, can be estimated by a stochastic process called *Monte Carlo Integration*. This is true for every definite integral of a function $f(x)$:

$$\int_a^b f(x)\ \partial x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}, \tag{2.21}$$

Monte Carlo Integration

where $p(x_i)$ is the probability to produce a sample x_i. Any Probability Density Functions (PDF) p can be used, but having a function that scales with f yields a faster convergence. Using such an improved distribution function is called *Importance Sampling*. However, at this point it is not clear how to select a sample based on a PDF. A common approach is the *Inverse CDF Method* also called *Inverse Sampling*. As the name suggests, we need a Cumulative Density Function (CDF) $F(x)$, which is given by:

$$F(x) = \int_c^x p(x)\ \partial x, \tag{2.22}$$

for a given PDF $p(x)$, assuming that the probability is defined on the interval $[c, d] \in \mathbb{R}^2$ with $c < d$. Hence, a value ξ of the CDF defines the probability to draw a sample with a value smaller than x for any $x \in [c, d]$, see Equation (2.23). When $p(x)$ is a valid probability density function, ξ will be in the interval $[0, 1]$. The idea of inverse sampling is to invert this function, take samples of uniform distribution between zero and one, and generate samples $x \in [c, d]$ using that inverse function F^{-1}:

$$F(x) = \xi \tag{2.23}$$

$$\Rightarrow \qquad x = F^{-1}(\xi)\,. \tag{2.24}$$

The generated x_i are distributed based on $p(x)$. More samples are generated in areas, where $p(x)$ is large, and fewer samples are generated, where $p(x)$ is low. This assumes that the inserted values ξ_i have been distributed uniformly, e.g., generated by an external random number generator available on any platform[6]. The described method also assumes that the PDF can be integrated and that the resulting CDF can be inverted analytically. If that is not the case, the integration can be done numerically. The inversion is then replaced by a binary search in the tabulated CDF.

After describing how to create suitable random samples for importance sampling, we can continue with the light transport simulation. Applying importance sampling to the rendering equation (2.12) gives:

$$L(\mathbf{x}, \boldsymbol{\omega}_o) \approx L_e(\mathbf{x}, \boldsymbol{\omega}_o) + \frac{1}{N}\sum_{i=1}^{N}\frac{f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, L(\mathbf{x}, \boldsymbol{\omega}_i)\cos\theta_i}{p(\boldsymbol{\omega}_i)}, \tag{2.25}$$

where N is the number of sampled incident ray directions $\boldsymbol{\omega}_i$ and $p(\boldsymbol{\omega}_i)$ is the probability of selecting those directions. While it is known that the PDF p

6 The quality of the produced random number distributions varies depending on the system and the implementation.

should be similar to the function in the numerator [Vea97], the selection of an optimal PDF – and thereby, the selection of an optimal set of samples – is still focus of active research, because the product in the numerator is unknown and often the PDF is based on $f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \cos\theta_i$ only, ignoring the incident radiance. Additionally, the selected sampling strategy has a very high influence on the variance in the result and thereby the convergence behavior.

For path tracing, only one random path is selected at each bounce. Hence, the number of samples N is large only during the ray generation (0^{th} bounce). For the following events, only one sample $\boldsymbol{\omega}_i$ is drawn ($N = 1$) and weighted by the density $p(\boldsymbol{\omega}_i)$, which further simplifies Equation (2.25). However, since this sample direction is created randomly, an other direction will be selected when a similar path is created. In the limit, all integrals will be estimated.

Similar to ray tracing, the direct light is evaluated at each hitpoint. This evaluation is often referred to as *next event estimation*. When multiple light sources or area light sources are present in the scene, it is also common to select a random source or a random position on the area source for the next event estimation.

For most real world materials, different types of light paths need to be evaluated. Therefore, one of the events is selected randomly. Glass for instance reflects and refracts incident light. The probability of reflection, which is defined by the Fresnel equations, is used as probability to continue with the reflection ray. Otherwise, a refracted ray is traced. Random decisions are also applied for mixed materials (see Equation (2.17)). Based on the diffuse amount, a ray is cast into a random direction of the upward hemisphere. Otherwise, the specular part is sampled for an outgoing direction or the ray is stopped in case of absorption.

Path tracing is said to converge to the correct solution after a sufficiently large number of iterations – actually after an infinite amount of samples per pixel – and therefore, it is called a *consistent* algorithm. Path tracing is also an *unbiased* algorithm. This means, that the expected error of the estimation is zero, independent of the number of iterations. Maybe more intuitively, an unbiased algorithm has no systematic error. The chance of overestimating and underestimating the integrand is equal and therefore, the error in an image is just noise, that can be reduced by increasing the number of samples. In contrast, it is possible to achieve faster convergence or less noise by accepting a systematic error. In this case the approach is said to be *biased*. When the estimator produces a bias that vanishes in the limit at $N = \infty$, the approach can still be *consistent* when it converges to the right solution. Progressive photon mapping, introduces in Section 2.6.6, is an example of a biased but consistent approach.

2.6.3 *Light Tracing*

As stated earlier, ray tracing and path tracing simulate the light transport in a backward manner, i.e., we are tracing *view rays* from the camera towards the light sources. It is also possible to reverse this strategy once more, which results in a *Light Tracer*. Therefore, Dutré proposed a particle-based approach that creates *photons* at the light sources and emits them in form of *light rays* depending on the intensity distribution of the particular light sources [DLW93]. After hitting a surface, the direction of the next path segment is chosen randomly, similar to path tracing. The flux carried by the photon can be reduced to account for absorption. Alternatively, the flux can be left con-

stant by keeping track of a survival probability and stopping based on random chance. At each bounce, a next event estimation is applied. In light tracing, this is a connection to the camera, where the visible radiance contribution is added to the relevant pixel.

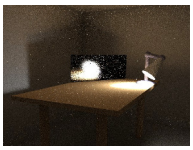
This technique is better suited to generate caustic paths, compared to path tracing, but shows problems with highly specular surfaces. However, since many of the next event estimations do not connect to a valid pixel of the image, a certain amount of computational effort is wasted.

2.6.4 *Bidirectional Path Tracing*

To utilize the advantages of light and path tracing, both techniques can be combined to Bidirectional Path Tracing (BPT), as introduced by Lafortune and Williams [LW93]. Here, a view ray for a pixel and a light ray from a selected source are cast at the "same time". The vertices of both paths are connected by shadow rays and the radiance contribution is added to the initial pixel of the view ray. In the context of bidirectional path tracing, Veach investigated the combination of different sampling strategies [VG95]. Depending on the size of the light source and the roughness of the material the light is reflected in, it can be better to generate reflected rays by sampling the BRDF of the material or the intensity distribution of the light source. To allow for a combined sampling that favors the most promising strategy, he proposed *Multiple Importance Sampling*.

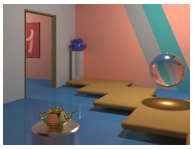
2.6.5 *Metropolis Light Transport*

A rather untypical approach to rendering, called Metropolis Light Transport (MLT), was presented by Veach and Guibas [VG97, PJH16]. Whereas the previous Monte Carlo methods rely on an independent sampling, MLT applies a correlated sampling. The technique is able to deal with difficult scenes, where important light paths are hard to find by previous techniques. Paths with a high contribution to the image, that have been found by chance, are mutated to create similar paths. Thereby, the path space is explored locally in a more guided manner, which increases the sampling rate and reduces the variance in difficult, otherwise noisy areas. Even though MLT is able to produce very good results in difficult scenarios, simpler approaches can show a faster convergence in easier scenes. Additionally, MLT is difficult to implement in comparison to the other mentioned approaches. Using MLT for rendering video sequences also required special strategies to ensure temporal coherence.



Courtesy of Veach and Guibas [VG95]

Courtesy of Veach and Guibas [VG97]



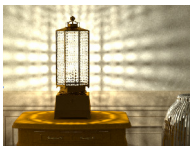
Courtesy of Jensen [Jen96]

2.6.6 *Photon Mapping*

In 1996, Jensen presented a bi-directional particle-based approach called Photon Mapping (PM) [Jen96, Jen01]. In the first of two passes, a large number of photons are created and traced similar to the light tracer. Instead of projecting photons directly into the image, they are stored in spatial data structures, called photon maps, as soon as they hit a diffuse surface or when absorbed. These data structures are usually realized by kd-trees or spatial hashmaps. The photons are stored along with their incident directions in one of two photon maps: one high resolution map for caustics and one with lower resolution for the remaining illumination. In case of a diffuse or specular event,

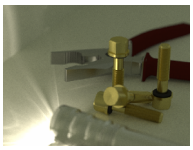
the tracing continues into the reflected or refracted direction. The particular case is again selected randomly based on the material properties at the hitpoint.

In the second pass, a (distributed) ray tracer or a modified path tracer is used to compute the radiance at every pixel by evaluating the photon maps. A *density estimate* (also called *radiance estimate*) is performed at the first diffuse event of the view ray. Therefore, the photons within a certain radius around the hitpoint are queried, which yields the approximated irradiance at that location. Because of this radius, PM is not unbiased anymore. The direct density estimate can be replaced by *final gathering* to sample the incident radiance from the entire hemisphere to achieve competitive results. Even with final gathering enabled, the caustic map is evaluated directly to be able to show caustics: that can be reproduced very efficiently in PM.



Courtesy of Hachisuka *et al.* [HOJ08]

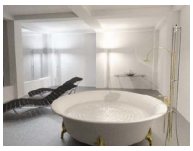
Like all methods above, PM can be computed in a progressive manner to arbitrarily increase the number of samples and thereby continue the simulation until visible artifacts have disappeared. In PM this is extremely useful as the large number of photons, created in the first pass, are distributed across multiple iterations, which also avoids memory constraints on the photon maps [HOJ08, HJ09]. It can be shown that (stochastic) progressive PM converges towards the correct solution by simply reducing the query radius for the density estimates with each iteration [KZ11]. Hence, this progressive simulation is consistent and easy to implement when a standard photon mapper is already available.



Courtesy of Hachisuka and Jensen [HJ09]

Photon mapping is well suited for caustics and translucent materials, or specular-diffuse-specular paths in general, and often faster than pure Monte Carlo methods. However, it is rather inefficient in diffuse lighting and tends to have longer computation times. A combination of PM and BPT is also possible. Georgiev *et al.* presented Vertex Connection and Merging (VCM), currently one of the best Monte Carlo approaches available [Geo+12]. By reformulating photon mapping as a bidirectional path sampling problem, they have been able to make both techniques compatible with each other. VCM adaptively weights both methods using a heuristic and thereby reaches renderings of higher quality within the same computation time. The same combination was published by Hachisuka *et al.* at the same time [HPJ12].

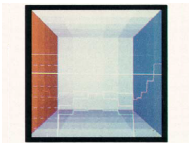
Courtesy of Georgiev *et al.* [Geo+12]



Courtesy of Hachisuka *et al.* [HPJ12]

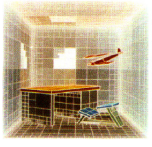
2.6.7 *Radiosity*

Another classic approach to compute light transport solutions is *Radiosity* introduced by Goral *et al.* [Gor+84]. It is a method based on finite elements. The general idea is to subdivide the scene into small surface elements, called *patches*, and then to compute the transfer between those patches. In the beginning, the scene was assumed to be completely diffuse. Thereby, all patches are considered to have a constant radiance. In that case, the amount of flux transferred between two patches can be precomputed into *form factors*. Then, the radiosity of a sender patch B_j can be described by the sum of its self emission E_j (note that this is not irradiance) and the reflected received radiosity B_i of all other patches:



Courtesy of Goral [Gor+84]

$$B_j = E_j + \rho_j \sum_{i=1}^{n} B_i F_{ij},$$



Courtesy of Hanrahan *et al.* [HSA91]

where ρ_j is the diffuse reflectance coefficient and F_{ij} the form factor between patch i and j. The equations for all pairs of patches can be used to set up a lin-

ear system. After solving, a view-independent solution for the light transport problem is found. However, this explicit solving works only for simple scenes because of the size of the system matrix. Therefore, the system is solved numerically, e.g., by progressive radiosity [Coh+88]. To reduce the number of transfers to compute for each patch, hierarchical radiosity was developed [HSA91].

Linear Gouraud interpolation can be used to get a smooth visualization of the result. However, it is more common to combine radiosity with ray tracing-based technique. The tracing is able to produce high frequent details in direct illumination and direct shadows, whereas radiosity provides indirect lighting. It is also straightforward to implement *final gathering* to improve the quality compared to a direct visualization [LTG93]. Final gathering also allows to evaluate different BRDFs to get plausible non-diffuse material effects.

Radiosity itself can be extended to non-diffuse materials [Rit+12]. Therefore, the directional distribution of incident and exitant light needs to be stored for each patch, e.g., in discrete bins.

Starting with the work of Goral *et al.*, Radiosity became the focus of a very active period of research [Rit+12]. It was also the foundation for many works in the early years of AR. Over time, Monte Carlo-based methods became more and more popular. Not only because of the required meshing step in the beginning, but also due to the form factor computation that requires ray tracing for visibility tests anyway. Furthermore, radiosity is by design a biased technique, not able to produce a correct solution even for perfect diffuse scenes.

2.7 Real-time rendering

The techniques covered before, mostly focus on the correct simulation of the light transport. This usually involves a high amount of computational effort and thereby results in long computation durations. In the extremely large field of real-time rendering, we try to reformulate and simplify the transportation problem to be able to compute plausible results at interactive rates. There are varying definitions of *interactive* and *real-time* performance. Here, we consider refresh rates of at least 5 Hz as interactive and 30 Hz to be real-time. However, in general it depends on the application. In some scenarios one update per second can be sufficient for interaction, whereas in VR for instance 60 Hz are barely enough to provide a smooth motion.

Many real-time approaches relate to the basic techniques discussed above, but the performance constraints require simplifications. Thus, the resulting light transport solutions are biased and differences to the correct results are visible to the naked eye. Compared to early (fixed-function) rendering pipelines, more recent engines rely on Physically-based Rendering (PBR). Instead of shading surfaces based on some light and material parameters, the participating scene elements are defined based on real-world quantities. The lighting and shading solely rely on the laws of physics. However, due to simplifications, certain aspects of light transport are usually approximated or even ignored. When the resulting renderings are still able to convince the users, the approaches are called *plausible*.

Courtesy of Lischinski *et al.* [LTG93]

All following parts of this thesis rely on basic and often more advanced knowledge of real-time rendering and especially about rasterization. There are various text books available on this topic. This includes for instance details on the rasterization pipeline, programmable shader, texture mapping, shadow

maps and various other real-time algorithms. As one first reference, I recommend the *Real-time Rendering Book* of Akenine-Möller *et al.* [AHH08].

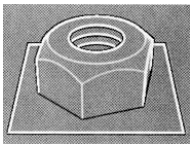
2.7.1 *Deferred Shading and Tiled Rendering*

The concept of *Deferred Shading* goes back to Saito and Takahashi [ST90]. They proposed to render the scene in a *geometry pass* to store all required geometry attributes in a *Geometry Buffer* (G-Buffer). This buffer contains the positions, normals and material properties – visible in the current view of the camera – per pixel. After the G-Buffer is created as intermediate result, further operations like shading can be computed as a post-processes and independent from the geometry complexity. For shading, this post-process is also called *lighting pass*, where we iterated over the light sources and accumulated their contribution.

Therefore, light is usually assumed to be emitted from sources with a finite range of influence. Even though the irradiance at a receiver surface decreases with the squared distance from the emitting light source, the irradiance will not get zero. By limiting the influence range of lights, the computation effort is reduced, but a bias is introduces. Hence, a point light can be seen as spherical volume with a fixed radius. A Lambertian emitter could be represented by a hemisphere and a spot light by cone. An easy approach to evaluate the contribution of each light is to use rasterization to render proxy-geometries, i.e., spheres in case of points lights. With enabled depth tests but disabled depth writing, only visible and not occluded parts of the proxy-geometries reach the fragment shader. For each fragment, the G-Buffer is read to reconstruct the scene surface visible at that pixel and the light contribution of the current light is computed and accumulated in the frame buffer. Note, that this decouples the geometry complexity of the scene from the lighting complexity. Hence, more lights can be processed than with conventional rasterization.

A related but different approach is *Deferred Lighting* or later called *Light Pre-Pass* [Eng09]. Here, the G-Buffer is reduced to attributes required for computing the irradiance per pixel. In a second pass, the pre-convolved diffuse and specular irradiance are computed and stored for each pixel. Eventually, in a third pass, which requires to render the geometry a second time, the shading is computed based on the material and the irradiance computed in the former step. However, this technique is not further discussed or used in this thesis, as no practical performance gain can be expected [Lau10].

Focusing on deferred shading again, there is one obvious problem: Every time a light influences the shading of a pixel, the G-Buffer is read to fetch the attributes required for shading, which leads to bandwidth problems. The game development community [BE08, And09, Lau10] addressed this problem by a new technique called *Tiled Deferred Shading*. After filling the G-Buffer, a screen space grid is constructed, where each cell, or *tile*, is of fixed size, e.g., 32×32 pixels. The screen space extents of each light source is tested for intersection with the grid cells. In case of an overlap, the light is appended to a list per cell. In the lighting pass, each pixel of the G-Buffer is read only once to reconstruct the surface properties. Then, all lights in the corresponding list of that pixel are iterated for shading. This not only reduces the bandwidth utilization for reading, but also writing, as the accumulation is done at floating point precision in the registers and the result is written only once after finishing the iteration. Since the pixels of each tile iterate over the same lights, thread coherence is high, which fits well to the GPU architecture. This ap-



Courtesy of Saito and Takahashi [ST90]

Courtesy of Andersson [And09]

proach also has drawbacks, mostly inherited from deferred shading. First, the G-Buffer requires a relatively high amount of memory, which is not available on all platforms. Second, because of the G-Buffer, anti-aliasing is not supported in a straightforward manner. Additionally, the most important issue is the lack of support for handling transparency.

However, all those problems are easy to handle by traditional rasterization, which is called *Forward Shading* in contrast to the deferred methods. The idea of tiled rendering can be directly applied to forward shading [OA11], as proposed by Olsson and Assarsson. This also decouples geometry from light processing by simply iterating over the list of lights within the fragment shader. The scene is rendered only once and all the benefits of tiled deferred rendering are also true for forward shading. Anti-aliasing, muli-sampling and transparency can be handled as without tiling. To deal with pixels that are shaded multiple times due to primitives that occlude already shaded geometry, a *z-prepass* can be used to fill the depth buffer before actually shading the scene. One could argue, that the overhead for submitting all draw calls twice, the additional vertex transformations and potentially many more operations like tessellation can easily exceed the gain in performance by avoiding overdraw, depending on the scene. But a z-prepass also allows to reduce the number of lights per tile as the minimum and maximum depth within the tile can be used to reject light sources that have no influence. The resulting technique is also known as *Forward+* [HMY12, McK12] and currently one of the preferred techniques as it provides high flexibility in terms of types of lights and BRDFs.

Olsson *et al.* suggests to subdivide the per tile frustums, and thereby the lists, also in the viewing direction to further reduce the light computations per pixel [OBA12a, OBA12b]. Therefore, they use the depth information provided by the G-Buffer or the z-prepass to cluster fragments per tile, based on their position and normal. Hence, lights can be rejected based on multiple depth intervals, instead of one in forward+. The normal also allows to apply a backface culling on a per-cluster basis.

Courtesy of Harada *et al.* [HMY12]

Courtesy of Olsson *et al.* [OBA12a]

Courtesy of Olsson *et al.* [OBA12b]

Courtesy of Keller *et al.* [Kel97]

Courtesy of Ritschel *et al.* [Rit+08]

Courtesy of Holländer *et al.* [Hol+11]

2.7.2 *Instant Radiosity*

The Instant Radiosity (IR) technique, presented by Keller, allows to simulate diffuse light transport in a two-pass process [Kel97]. First, photons are cast from light sources, similar to PM. Instead of storing these photons in a map, each photon is interpreted as Virtual Point Light (VPL) that emits (indirect) light into scene. In the second pass, each VPL is rendered using the rasterization pipeline, where the scene geometry is drawn and shaded by the currently active VPLs. The contributions of the lights are accumulated to the final image, which can also be refined in a progressive manner. The rasterization replaces the density estimation of PM. Since each VPL is used to illuminate all pixels, the amount of perceived noise is much lower compared to Monte Carlo methods, as the traced paths are not independent for each pixel [Dac+14]. However, due to the lighting of the scene and the shadow map generation for each VPL, the scene geometry has to be rendered several times.

To deal with *overdrawing* during shading, the second pass makes use of deferred shading or the more recent tiled shading (see above). Hence, the illumination by VPLs is reduced to a screen space operations, where relevant lights are iterated per pixel.

To reduce the overhead during the shadow map generation, Ritschel *et al.* proposed Imperfect Shadow Maps (ISM) [Rit+08]. They assume that the visibility of a VPL can be approximated by a shadow map of low resolution. This is justified by the low frequency nature of indirect light. When a high number of VPLs and their corresponding shadow maps are used to shade a single pixel, the influence of each VPL is limited and indirect shadows get smooth by many overlapping hard shadows, cast by the individual point lights. The key idea of ISM however, is not only the low resolution, but also to represent the scene as point cloud. A different random set of points are splatted into each of the small shadow maps. This naturally results in holes that can be filled by push-pull steps. Instead of splatting random points into random shadow maps, Holländer *et al.* suggest to use a hierarchy and to select the appropriate node for each pixel of the shadow map [Hol+11]. Essentially, they compute a graph cut for each shadow map in parallel, which yields better quality in approximately the same time because holes are avoided and push-pull is not necessary. In dynamic scenes, the cut of the last frame can be refined, which is another benefit of this approach referred to as *ManyLoDs*.

Courtesy of Dachsbacher and Stamminger [DS05]

The positioning of the VPLs was initially defined by a Central Processing Unit (CPU)-based ray tracing. Instead, the first bounce of photons emitted from a primary light source can also be computed by rasterization [DS05]. Therefore, a cube map is rendered at the position of the light and stores position, normal and material properties per pixel, similar to a G-Buffer. This map is called Reflective Shadow Map (RSM) and can be sampled by importance to create VPLs for (first-bounce) indirect light.

Courtesy of Segovia *et al.* [SIP07]

A number of extensions to IR have been presented. Bi-directional IR allows to generate VPLs depending on the position of the camera and thereby increases the efficiency of the approach especially when multiple bounces are computed [Seg+06]. Segovia *et al.* also presented Metropolis sampling for IR [SIP07] to handle scenes with more difficult light paths. VPLs have originally been proposed as Lambertian emitters on diffuse surfaces. Several approaches tried to generalize the idea to non-Lambertian materials [DS06, Rit+08, Haš+09]. The latter work, presented by Hašan *et al.*, also suggests to use Virtual Spherical Lights (VSLs) to avoid singularities in corners and concave edges that are typical for VPLs. Dong *et al.* clustered VPLs based on their position and orientation to get Virtual Area Lights (VALs), that allow to apply efficient soft shadows techniques for evaluating the visibility [Don+09].

Courtesy of Hasan *et al.* [Haš+09]

While every VPL is potentially used to shade every pixel on the screen, variants of IR referred to as *Many-Light Approaches*, try to reduce this linear-time complexity [Wal+05]. By creating light hierarchies with VPLs as leaf nodes, distant pixels can be illuminated by inner nodes without notable difference. This technique is called *Lightcuts*, as the shading for each pixel depends on a cut through the hierarchy of thousands to millions of point lights.

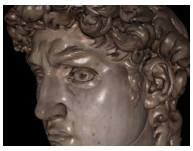
Courtesy of Dong *et al.* [Don+09]

More details and further extensions can be found in the report on many-light techniques by Dachsbacher *et al.* [Dac+14] and also in the report by Ritschel *et al.* [Rit+12].

Courtesy of Walter *et al.* [Wal+05]

2.7.3 *Micro-Rendering*

Computing Global Illumination (GI) solutions based on a point-based representation of the scene is the goal of another group of rendering methods. Point-based rendering techniques, without considering GI, go back to Levoy and Whitted [LW85]. One of the most famous point rendering systems is



Courtesy of Rusinkiewicz and Levoy [RL00]

QSplat, which supports millions of points in a hierarchy of bounding spheres [RL00]. The inner nodes represent the data of their child nodes by storing an averaged position and radius, an approximate normal and a normal cone as well as color. By selecting and splatting the inner nodes instead of all leaves, the system is capable of efficiently splatting only the nodes of proper size with respect to the pixel. This is also the basic idea of many more recent approaches. Similar to these inner nodes, the leaf-nodes can also have an orientation and a spatial extent. In this case, they are called *surfels* [Pfi+00, Zwi+01], i.e., small oriented disks that can overlap to guarantee a hole-free rendering. An overview of these early techniques, also containing some first GPU techniques, can be found in the report of Kobbelt and Botsch [KB04].

One of the first techniques that made use of surfels for GI effects was the dynamic ambient occlusion and indirect lighting technique by Bunnell [Bun05]. He replaced triangles or quads by disks and approximated ambient occlusion by accumulating the shadowing of other surfels based on form factors. Without considering the visibility between emitting and receiving surfels, multiple iterations are used to compensate double shadowing. The author also showed diffuse indirect lighting by exchanging the form factor-based occlusion with a disk-to-disk radiance transfer. Despite the assumptions made, the presented results are plausible. To reduce the computational complexity, Bunnell proposed to use a hierarchy, by grouping neighboring surfels. During the traversal, he used a heuristic to select the hierarchy level based on the distance and the radius of the disks.

Inspired by this work, Christensen developed a CPU-based renderer that relies on a point-cloud representation of the scene [Chr08] to compute a GI solution, the method works in three passes. First, the geometry is sampled to generate the surfels, which are then clustered and added to an octree. The inner nodes of the tree store an approximation of their child nodes compressed in Spherical Harmonics (SH) (see Section 2.7.6). In the second pass, the surfels (leaf nodes) are shaded by computing direct illumination and inner nodes are updated by pulling radiance up the tree. Third, for every pixel on the screen, a small image – later called *Micro Buffer* – is rasterized to gather irradiance from all locally visible surfaces. Therefore, one of three methods, depending on the distance of the receiver point, is used: ray tracing for close emitters, splatting a single surfel at medium distance or evaluating a node of the octree for distant geometry. Convoluting the discretized irradiance with the BRDF at the pixel than yields a correctly illuminated surface. The method achieves high quality rendering 4 to 10 times faster then ray tracing, which is achieved by applying the QSplat algorithm [RL00] of Rusinkiewicz and Levoy. Here, QSplat is used to select the proper Level of Detail (LoD) in the octree, i.e., a node that fits the solid angle of the corresponding pixel in the micro buffer. If multiple nodes are valid candidates, which can happen because of multiple layers of occluding geometry, a simple depth buffer is used to select the closest.

Courtesy of Bunnell [Bun05]

The term *Micro-Rendering* was proposed by Ritschel *et al.* [Rit+09]. Their technique is based on the point-based GI by Christensen and uses the GPU to perform the rasterization to all *Micro-Pixels* in parallel to achieve interactive frame rates. Instead of using an octree, they compute a binary-space partitioning of the surfels and store a full binary tree that can be traversed in constant time without storing pointers. Inner nodes store a minimum bounding sphere, enclosing all child nodes, and a cone that captures their normal variation. Similar to Christensen, a micro-buffer is used for final gathering. For each micro-buffer, the tree is traversed in depth-first order starting at the

Courtesy of Ritschel *et al.* [Rit+09]

root. If the solid angle of the node gets smaller than the solid angle subtended by the mirco-pixel it would be projected to, the correct node for that pixel is found and the next node (right or up in the tree) is processed. Otherwise, the depth-first search is continued with the left child. In case the traversal reaches a leaf node that is still too large, the node is stored in a list and ray tracing is used afterwards for all empty pixels to determine accurate intersections with the nodes in the list. This yields a micro-buffer filled without holes. Instead of simply convoluting the gathered irradiance with the BRDF, the authors applied a warping to the projection of the micro-buffer, based on the BRDF. This is similar to importance sampling, as important reflection directions get more pixels in the micro-buffer than less likely directions. To deal with more complex reflection models, the node indices, stored in the micro-buffer, can be used to fetch the corresponding surfel and compute a more complex reflection then a simple diffuse bounce. As an alternative implementation, *ManyLoDs* can be applied to find a graph cut per micro-buffer. By applying only a few iterations to update the graph cut of the previous frame, an improved performance can be achieved while getting similar results [Hol+11].

2.7.4 *Image-based Rendering*

To be able to present GI solutions in real-time, the scenario can be restricted to conditions in which the computation or at least the presentation can be performed in real-time. The following approaches consider different parts of the scene definition to be static, i.e., that the objects, their materials or the light condition does not change over time. Because these elements stay constant throughout the session, it is possible to pre-compute parts of the light transfer in advance.

For Image-based Lighting (IBL), the light condition is assumed to be constant over time. The environment that illuminates an object is also considered distant. Hence, the radiance $L(\mathbf{x}, \boldsymbol{\omega}_i)$ in the rendering equation (2.12) can be simplified to a directional function $L(\boldsymbol{\omega}_i)$, which is independent of $\mathbf{x}$.

Each ray that is cast for gathering the incident light – and that is not intersecting the object itself – can be evaluated solely by the direction of the ray. The result for all of these ray casts can be stored in texture, called *Environment Map* or *Light Probe*, which is indexed by the direction of $\boldsymbol{\omega}_i$. One way to create such a map, is to render an omnidirectional image from the position of the object to illuminate. However, it also common to use a photographed map that shows a similar scene, as reflections and indirect illumination do not need to be exact to create a plausible result. Here, it is assumed that users do not realize an error if the reflections roughly match their expectations. Note, that IBL is actually a more general term for rendering using an image-based representation of the environment (see Section 4.1.2). However, IBL is commonly used as equivalent for the special case that uses light probes or environment maps as input.

For rendering, each pixel of the environment map is considered a directional light source and accumulating their contribution leads to a numerical solution of the rendering integral. For IBL it is common to use importance sampling or to subdivide the map into regions of different size but same radiance to improve performance by concentrating on the directions with the largest irradiance [CD01, Aga+03, ODJ04, Hav+05a]. Sampling leads to a discrete set of light sources that can be used in the rasterization pipeline. Shadow maps, e.g., by applying the ManyLoD-scheme, can be used to determine visibility. More

Courtesy of Ostromoukhov *et al.* [ODJ04]

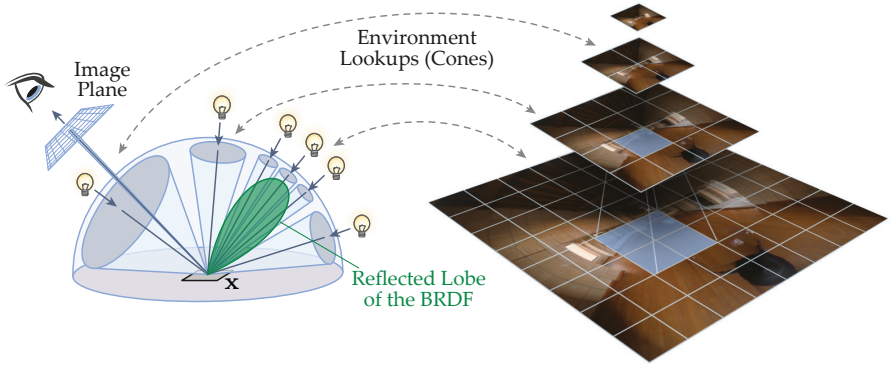


Figure 2.5: GPU Importance Sampling
Illustration of the BRDF-driven importance sampling for Monte Carlo rendering on the GPU. Image courtesy of Colbert *et al.* [CK07].

details on IBL can be found in the HDRI Book [Rei+10]. Discussions about rendering using pre-filtered environment maps can be found in the course notes of Jan Kautz [Kau06].

For traditional real-time rendering, these light probes have been used for simulating perfect specular reflections and refractions. The so called *Reflection Mapping* technique, also Environment Mapping (EM), goes back to Blinn and Newell who rendered the first object, a Utah Teapot, with environment mapping [BN76].

Courtesy of Blinn and Newell [BN76]

This limitation on smooth materials can be overcome by applying a low-pass filter. In 1984, Miller and Hoffman [MH84] filtered the probe to enable diffuse illumination. For each texel the hemisphere pointing into the direction of that texel is integrated:

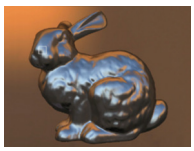
$$D(\boldsymbol{\omega}_o) = \int_{\mathcal{H}_+} L(\boldsymbol{\omega}_i) \, \frac{\rho}{\pi} \, \overline{\cos}\,\theta \; \partial\boldsymbol{\omega}_i,$$

where θ is the angle between the direction of the texel and the direction of $\boldsymbol{\omega}_i$. The diffuse BRDF ρ/π can be factored out. For evaluating the diffuse illumination, the surface normal $\mathbf{n} = \boldsymbol{N}(\mathbf{x})$ is used for the lookup: $D(\mathbf{n})$. Applying a different convolution function allows to pre-filter the light probe for glossy reflection, e.g., using Blinn-Phong with shininess n:

Courtesy of McAllister *et al.* [MLH02]

$$S(\boldsymbol{\omega}_o) = \int_{\mathcal{H}_+} L(\boldsymbol{\omega}_i) \, \frac{n+1}{2\pi} \, \overline{\cos}^{\,n}\psi \; \partial\boldsymbol{\omega}_i.$$

This time, the map is queried by the reflection vector $\mathbf{r}$ instead of the surface normal: $S(\mathbf{r})$. Miller and Hoffman also suggested to store environment maps in six cube face images, the well-known *Cube Map*. To support spatial variations in glossiness, the MIP levels can be used to store pre-filtered maps for different shininess values [MLH02].



Courtesy of Colbert and Křivánek [CK07]

IBL is also referred to as *Natural Illumination* and when used with a captured real-world light probe, it is one of the basic concepts in AR.

In Chapter 7, we use a more general and GPU-accelerated solution by Colbert and Křivánek [CK07, KC08]. The idea is illustrated in Figure 2.5. To compute the shading of a point on the surface $\mathbf{x}$, we need to sample the entire hemisphere above $\mathbf{x}$ and weight incident light by the BRDF at the surface point. For glossy materials, the most important directions are those around the reflected view direction, here visualized by the green lobe. Because these

Courtesy of Křivánek and Colbert [KC08]

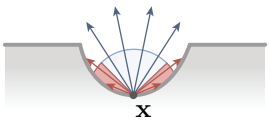


Figure 2.6: Ambient Occlusion
Ambient occlusion describes the amount of incident light at a surface element $\mathbf{x}$, that is blocked by geometry of the local neighborhood.

directions are important, we place many samples there and fewer samples in other directions. The idea is now to use different filter levels, based on the importance, which maps to the mip-chain on the GPU. In areas with high probability, we use detailed levels. In areas with low probability, where we draw only few samples, we use coarser mip levels to get a low frequency approximation without missing light information. This technique allows to compute the shading of materials with arbitrary BRDF.

2.7.5 *Ambient Occlusion*

By considering the geometry of the objects in the scene to be static, the visibility between surface elements can computed in advance. A simple and well known method that makes use of this concept is AO [Mil94]. Here, self-occlusion or self-shadowing caused by concave details in an objects geometry is precomputed for each surface position $\mathbf{x}$. Since ambient light can be interpreted as a constant approximation of light that bounced multiple times in the environment, AO can also be seen as a very simple global illumination approximation. Therefore, we consider the binary visibility function $V(\mathbf{x}, \boldsymbol{\omega})$, which has the value zero, if there is a self-occlusion when looking from $\mathbf{x}$ into direction $\boldsymbol{\omega}$ and one otherwise. Integrating over the upward hemisphere above $\mathbf{x}$ yields the percentage of the incident light directions that are not blocked by other geometry, which Miller describes as *accessibility* of a surface (see Figure 2.6). Weighting with the cosine gives an attenuation factor that is used to scale the irradiance during shading and thereby an approximation for contact shadows:

$$AO(\mathbf{x}) = \frac{1}{\pi} \int_{\mathcal{H}_+} V(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta \; \partial\boldsymbol{\omega}_i.$$

Figure 2.7 shows an example. Assuming Lambertian materials and constant illumination from all directions, i.e., ambient light, AO provides the correct solution for direct lighting with shadows.

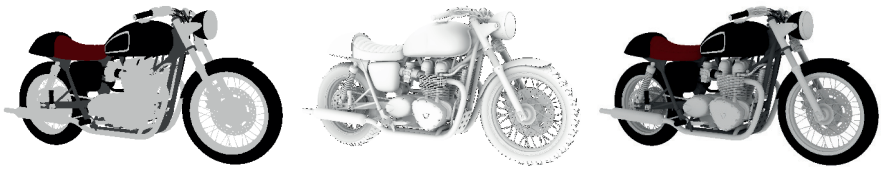


Figure 2.7: Ambient Occlusion: Cafe Racer
AO is used to approximate GI effects by applying a scalar factor to the computed shading. In the simplest case, as visualized here, the AO factor is simply multiplied to the diffuse reflectance coefficients, which results in plausible view-independent contacts shadows. Model courtesy of Maciek Ptaszynski.

To create a link to the rendering equation (2.12), we need to split the incident light from $\boldsymbol{\omega}_i$ and the visibility of the surface element at $\mathbf{x}$ from that direction. Hence, we need to consider the not shadowed radiance from all surfaces in direction $\boldsymbol{\omega}_i$ and their visibility $V^z(\mathbf{x}, \boldsymbol{\omega}_i)$, which is one only for the closest surface, i.e., the surface with the smallest distance z to the receiving surface element, and zero otherwise:

$$L_i(\mathbf{x}, \boldsymbol{\omega}_i) = \int_{z=0}^{\infty} L_i^z(\mathbf{x}, \boldsymbol{\omega}_i)\, V^z(\mathbf{x}, \boldsymbol{\omega}_i)\, \partial z = L_i^*(\mathbf{x}, \boldsymbol{\omega}_i)\, V(\mathbf{x}, \boldsymbol{\omega}_i)\,.$$

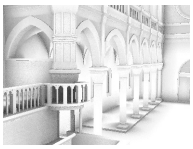
The rendering equation, neglecting the self-emission $L_e(\mathbf{x}, \boldsymbol{\omega}_o)$, then becomes:

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\mathcal{H}_+} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, L_i^*(\mathbf{x}, \boldsymbol{\omega}_i)\, V(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta\ \partial\boldsymbol{\omega}_i. \tag{2.26}$$

By now splitting the integral into a product, we get an incorrect but in some cases plausible approximation for quick renderings in movies and games:

$$L(\mathbf{x}, \boldsymbol{\omega}_o) \approx \int_{\mathcal{H}_+} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, L_i^*(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta\ \partial\boldsymbol{\omega}_i \cdot \frac{1}{\pi} \int_{\mathcal{H}_+} V(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta\ \partial\boldsymbol{\omega}_i,$$

where L_i^* can be evaluated more quickly by omitting shadow tests. This results in a lack of directional and hard shadows, as AO only depends on the geometry but not on the light condition. When shadow tests are not omitted, AO in combination with ambient light can be interpreted as approximation of indirect illumination and thereby as a very coarse GI approach. Adding the cosine to factors is optional and considered a stylistic choice, with no mathematical foundation.



Courtesy of Bavoil *et al.* [BSD08]

Over the years different implementations have been presented. Many of them are screen-spaced to allow for dynamic scenes (e.g., [Mit07, BSD08, Mit12]). Over time, the sampling strategy was improved to increase quality and performance while maintaining or reducing the number of samples. In the context of a master's thesis of Florian Bäthge, we investigated AO approaches for mobile devices [Bät15]. Since texture fetches are the main bottleneck in this case, only a low number of samples can be used.

Courtesy of Florian Bäthge [Bät15]

While AO describes the visibility for ambient illumination only, directional information is added in Screen-Space Directional Occlusion (SSDO) presented by Ritschel *et al.* [RGS09]. By considering the incident radiance of each sample, e.g., by sampling an environment map, the generated shadows become directed and colored. Furthermore, they treated samples that are classified to be occluders as indirect light sources. After estimating form factors, one bounce of indirect light is accumulated, which makes this approach a more plausible GI approximation.

Courtesy of Ritschel *et al.* [RGS09]

Note, that all techniques limit the distance of the occluders to a certain range, to deal with indoor scenes for instance, that would be dark otherwise. It is also common to influence the strength of the effect by user defined parameters.

2.7.6 *Precomputed Radiance Transfer*

We now combine the ideas of the last two approaches and assume distant, but not necessarily static, light as well a static scene geometry. While AO provides only a constant factor to model occlusions between surface elements,

Courtesy of Sloan *et al.* [SKS02]

Precomputed Radiance Transfer (PRT), presented by Sloan *et al.*, address the problem more generally [SKS02]. Besides the self-occlusions, their approach also handles diffuse interreflections and can be extended to deal with glossy reflections, caustics, and subsurface scattering for instance [Rit+12]. In this thesis, we focus on the diffuse lighting, which is the simplest part of the framework.

The core idea of PRT is to compute the integral of the rendering equation (2.12) in another domain. Therefore, a suitable basis is used to represent the input data in a domain where computations are easier to perform. In case the transformation into the new domain and performing the computations requires less time than the computations in the original domain, we can expect a performance gain.

In classical PRT, the selected basis functions are Spherical Harmonics (SH), an orthonormal basis over the sphere, analogous to the Fourier transformation over the spherical domain [SKS02]. For a point on the sphere given in polar coordinates:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \sin\theta\cos\varphi \\ \sin\theta\sin\varphi \\ \cos\theta \end{bmatrix},$$

the *real spherical harmonics* functions, denoted as y_l^m are defined as follows:

$$y_l^m(\theta,\varphi) = \begin{cases} \sqrt{2}\, K_l^m \cos(m\varphi)\, P_l^m(\cos\theta) & m > 0 \\ \sqrt{2}\, K_l^m \sin(-m\varphi)\, P_l^{-m}(\cos\theta) & m < 0 \\ K_l^0\, P_l^0(\cos\theta) & m = 0 \end{cases},$$

where P_l^m are the *Associated Legendre Polynomials* and K_l^m are normalization factors:

$$K_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-|m|)!}{(l+|m|)!}}.$$

The parameter l with $l \in \mathbb{N}$ is called the *band* index and defines the polynomial order. Furthermore, m with $-l \leq m \leq l$ is the *index* inside the band. Hence, l starts with 0 and contains only one basis function y_0^0, that can reproduce polynomials of degree 0, i.e., constant values. The next band with $l = 1$ has three basis functions, the one after that contains five basis function and so forth. Ramamoorthi and Hanrahan showed that for a not occluded diffuse illumination it is sufficient to use 9 basis functions, i.e., bands 0 to 2 [RH01b]. Since the associated Legendre polynomials are defined recursively, the basis functions can be precomputed for a discrete set of directions and stored in a cube map (see Section 5.2.4). It is useful to be able to enumerate the basis functions, e.g., to be able to store coefficients in linear vector:

$$y_i(\theta,\varphi) = y_l^m(\theta,\varphi) \quad \text{where} \quad i = l(l+1)+m.$$

To represent a spherical function, like the visibility $V(x,\boldsymbol{\omega}_i)$ or the incident radiance $L(\boldsymbol{\omega}_i)$, in the spherical harmonics basis, the function is *projected* onto the basis functions. As stated above, the SH basis is orthonormal. Hence, the following two properties apply:

$$\int_{\mathcal{H}_\pm} y_i(\boldsymbol{\omega})\, y_j(\boldsymbol{\omega})\ \partial\boldsymbol{\omega} = 0 \qquad i \neq j$$

$$\int_{\mathcal{H}_\pm} y_i(\boldsymbol{\omega})\, y_i(\boldsymbol{\omega})\ \partial\boldsymbol{\omega} = 1 \qquad \forall i.$$

Consequently, we can project the spherical input function, here $L(\boldsymbol{\omega}_i)$, onto each basis function y_i individually, which yields corresponding coefficients c_{li}:

$$c_{li} = \int_{\mathcal{H}_\pm} L(\boldsymbol{\omega})\, y_i(\boldsymbol{\omega}) \;\; \partial\boldsymbol{\omega}.$$

Using those coefficients, we get an approximated reconstruction of the input function by a simple linear combination:

$$L(\boldsymbol{\omega}) \approx \sum_i c_{li}\; y_i(\boldsymbol{\omega})\,.$$

The approximation is band-limited, which means that the original signal can be resembled more exactly with an increasing number of bands. In the limit, the original function is reproduced correctly. By reducing the computations to a certain number of bands, only the low frequency information are represented, while the higher frequencies are removed. In the case of diffuse illumination, we exploit that behavior to reduce the computational effort.

However, to compute the illumination we again consider the rendering equation augmented by the visibility function (see Equation (2.26)). In addition to the already projected incident light, the visibility and the cosine are also transformed into SH-space. This term, also referred to as *Transfer Function*, models the local response of the object to the incident light. Unlike the distance light, it needs to be computed for each receiver point $\mathbf{x}$. Usually, the coefficients are precomputed for all vertices v of the model, but storing coefficients in textures is also possible:

$$c_{vi} = \int_{\mathcal{H}_\pm} V(\mathbf{x}_v, \boldsymbol{\omega})\, \overline{\cos}\,\theta \; y_i(\boldsymbol{\omega}) \;\; \partial\boldsymbol{\omega} = \int_{\mathcal{H}_\pm} T_v(\boldsymbol{\omega}) \;\; \partial\boldsymbol{\omega}$$

$$T_v(\boldsymbol{\omega}) \approx \sum_i c_{vi}\; y_i(\boldsymbol{\omega})\,.$$

The most important reason to switch into a space with orthonormal basis is that the integral of the product of the functions, is guaranteed to equal the dot product of their projected coefficients:

$$\int_{\mathcal{H}_\pm} L(\boldsymbol{\omega})\, T_v(\boldsymbol{\omega}) \;\; \partial\boldsymbol{\omega} = \sum_i c_{li}\; c_{vi}.$$

For a diffuse BRDF, the radiance transfer then reduces to:

$$L(\mathbf{x}_v, \boldsymbol{\omega}_o) = \frac{\rho_d}{\pi} \sum_i c_{li}\; c_{vi}, \tag{2.27}$$

where the light emitted from vertex v is assumed to be zero. Figure 2.8 shows the resulting illumination of a diffuse motor bike in different light conditions. Note, that coefficients of the transfer function, c_{vi}, are the same for both renderings. Equation (2.27) is evaluated at each vertex and interpolated over the triangles. Instead of solving an integral for each pixel of the image, PRT requires only a few multiply-add operations per vertex to achieve global illumination features. Besides simple diffuse illumination, diffuse interreflections can be added by altering the transfer function to consider indirect light instead of the binary visibility. This only affects the pre-processing while the run-time performance is not influenced.

Another important property of SH functions is that they are rotationally invariant. This means, that the resulting coefficients are independent of the

Figure 2.8: Precomputed Radiance Transfer: Cafe Racer
In contrast to AO, visualized in Figure 2.6, provides PRT directional illumination in addition to the contact shadows.

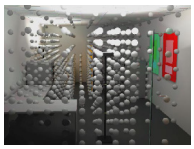
order of projection and rotation. A coefficient vector can be rotated by a linear function. However, it is not easy to implement the rotation efficiently [Gre03]. In our approaches, we avoid the rotation of coefficients by aligning the coordinate systems of the transfer function and the incident light. If the model to illuminate is rotated by the user, we re-sample the environment and thereby rotate the input data before projecting into SH basis.

The PRT framework also allows more complex light transfer simulations. Sloan *et al.* already showed glossy reflections that are handled by coefficient matrices rather than vectors, soft shadows of objects under rigid transformation, volumetric transfer to illuminate clouds as well as simple caustics [SKS02].

Spherical harmonics are not the only possible basis. The most common ones beside SH are wavelets [NRH03, Liu+04]. Special capturing devices have been developed to directly acquire coefficients in custom basis [Cal+13].

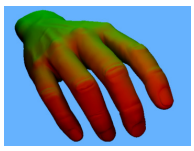
Courtesy of Ng *et al.* [NRH03]

References to more recent developments in the field of PRT can be found in the report of Ritschel *et al.* [Rit+12]. This includes alternate basis functions, support for BRDFs and dynamic objects. Practical details on the implementation of the SH projections and PRT, can be found in the notes of Robin Green and Peter-Pike Sloan [Gre03, Slo08].



Courtesy of Greger *et al.* [Gre+98]

For IBL, AO and PRT it is assumed that the scene is constant, but the distant lighting and the view position can be changed. This is inherited from the idea to use a light probe or an environment map, created at the position of the object of interest. Therefore, the radiance is expected to change slowly with respect to position. In many cases this assumption is not a bad one, compared to the effort of computing a correct solution, as radiance does change slowly in many settings. However, there are cases where the assumption is violated, e.g., close to shadows or at the boundary of the cone of a spotlight [Oat06]. In areas with rapidly changing radiance, a grid of multiple probes, *Radiance Volume*, can be used. Interpolation between these probes, and optionally incorporating gradients [Gre+98, WH92, Ann+04], can be used to address the issue. Note, that *Radiance Volumes* and *Irradiance Volumes* are storing different light quantities corresponding to their names. While a radiance volume represents the incident radiance from a certain solid angle $\boldsymbol{\omega}$ for all points and directions in the volume, irradiance volumes store the cosine weighted integral of incident radiance over the hemisphere pointing into the direction of that $\boldsymbol{\omega}$. However, they share the idea to provide light information in a grid that is interpolated for computing spatially varying illumination. Compressing the information using SH is an obvious extension of the idea, used for instance by Gibson *et al.* and Oat [Gib+03, Oat05]. Storing only a few coefficients with each grid cell yields a compact volumetric representation that can be used with PRT.



Courtesy of Annen *et al.* [Ann+04]

Courtesy of Kaplanyan [Kap09]

2.7.7 *Light Propagation Volumes*

A real-time technique to approximate first bounce diffuse global illumination was implemented in the CryEngine and presented by Anton Kaplanyan [Kap09]. As Light Propagation Volumes (LPVs) do not require any preprocessing stages, they support fully dynamic scenes including dynamic illumination, objects, materials and camera movements. Hence, it was adopted quickly in other engines too as it provided fast GI effects without increasing production effort. The basic concept of LPVs is to simulate diffuse light transport in a SH-based irradiance volume that spans the (currently visible) scene. It is thereby a finite-element method that works on a light field.

Before light is *propagated* through the volume, as the name suggests, the light needs to be *injected*. Motivated by instant radiosity, one or multiple RSM are used to generate secondary light sources (VPLs) on the directly illuminated surfaces. In general, it is possible to create those secondary lights differently. To inject a VPL, we consider the corresponding texel of the RSM as oriented surfel and project the reflected radiance of that surfel into SH basis. The contribution of the VPL is then added to the closest grid cell of the irradiance volume.

After the injection, the radiance of each cell is scattered to the direct neighbors in a forward propagation manner. However, a gathering scheme is usually implemented, as this maps better to the architecture of GPUs. The propagation is repeated multiple times depending on the resolution and the desired maximum travel distance.

Depending on the renderer, the LPV can be sampled during the per pixel shading or added in a deferred rendering pass. To reduce the memory footprint of the volume, only 2 bands, i.e., 4 coefficients are used. Nevertheless, the spatial resolution is rather low in most scenarios, which causes light bleeding through thin geometry because of trilinear interpolation. More details, including the handling of blockers can be found in the paper of Kaplanyan and Dachsbacher, who extend the method to cascaded LPVs [KD10].

2.7.8 *Voxel Cone Tracing*

Not long after the publication of LPVs, Crassin *et al.* came up with an approach that had the potential to replace LPV in real-time engines [Cra+11, Cra11]. Similar to LPV, Voxel Cone Tracing (VCT) does not require any preprocessing, but is able to compute indirect lighting without being restricted to low-frequency illumination. It is based on a sparse voxel octree that represents the geometry of the scene and that stores surface properties at different levels of detail within the octree hierarchy [Cra+09].

To achieve real-time performance in dynamic scenes, the octree representation needs to be updated on the fly. Therefore, the scene is rasterized orthographically at maximum octree resolution from the three main axes of the world coordinate system and the sparse data structure is constructed directly by traversing the tree for each fragment from top-to-bottom and subdividing nodes when needed. The opacity of each voxel is stored as floating point value per cell. After the voxelization, leaf nodes have an opacity of one or zero. Storing voxel nodes of size $2 \times 2 \times 2$ in combination with so called *bricks* of size $3 \times 3 \times 3$ that are stored in texture memory and contain surface properties like normal distribution and parameters for view-dependent BRDFs yields a sparse

Courtesy of Crassin *et al.* [Cra+11]

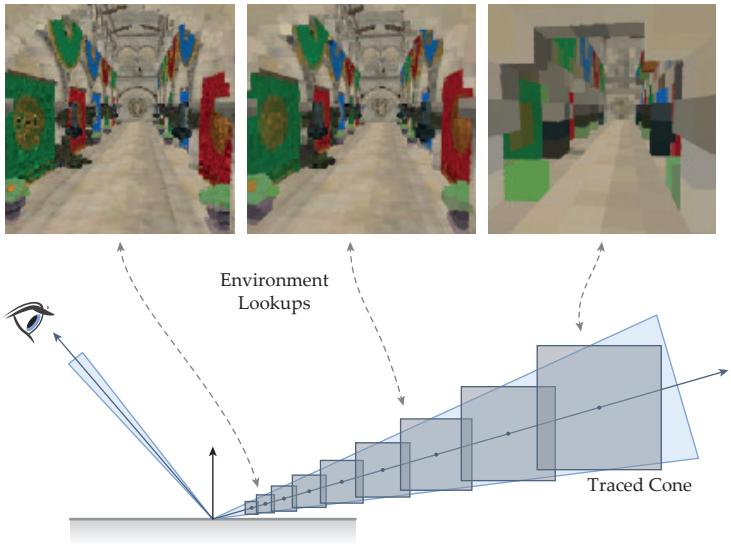


Figure 2.9: Voxel Cone Tracing
Cones are traced by sampling the voxel octree at different levels depending on the current cone width. Image courtesy of Crassin and Green [CG12].

but flexible data structure that also allows hardware interpolation within the bricks.

The algorithm consists of three steps: First, radiance is injected into the leaf level of the volume by rasterizing the scene from the perspective of the light and storing the incident radiance in the corresponding voxels, which later allows to simulate glossy reflections. In the second step, the data stored in the leaves is filtered into the higher levels of the hierarchy, which is similar to generating mip maps in a dense volume. That filtering step includes the computation of the opacity of inner nodes as average over their children. For the rendering, the scene is illuminated directly by the light sources and indirect light is queried by tracing secondary rays in the octree. At this point, the approach is highly similar to raytracing (see Section 2.6.1) but with one major difference: cones are traced instead of rays. These cones can be interpreted as a bundle of rays that allow to sample incident light from a larger solid angle, based on the aperture of the cone. In VCT, the idea is to select the sampling level in the octree based on the wide of the cone. Close to origin, the width of cone is small, therefore we are sampling at detailed levels or even the leaves. While tracing further along the ray, the width increases and lower resolutions of the hierarchy are selected for sampling with trilinear interpolation. Furthermore, we increase the step size during the tracing also dependent on the current width of the cone, which leads to larger steps, the further we are going (see Figure 2.9). This leads to a ray marching variant with an increasing step size that coincides with the size of the currently sampled voxel.

Eventually, the cone tracing is performed multiple times for every surface point directly visible to the camera to perform an approximated final gathering. A low number of large cones, usually five to eight, are traced for diffuse reflections. An extra cone with an aperture related to the roughness of the local surface is cast into the direction of the reflected view ray to sample the specular reflection. Varying the size of the cones depending on the BRDF leads to scheme visualized in Figure 2.5, which will be used later in Chapter 7.

Similar to LPV, VCT also suffers from light bleeding through thin geometry, which is the main drawback of the approaches. A bottleneck of VCT is the voxelization. Dense voxel structures require a large amount of memory and sparse octrees need more time to build. For more detail on the data structure, sampling and filtering, I refer to the well written paper [Cra+11] and Crassin's thesis [Cra11].

3

FUNDAMENTALS IN AUGMENTED AND MIXED REALITY

While the definition of Augmented Reality and the connection to VR as well as some examples and typical applications have been discussed in the introduction, this chapter will cover fundamental concepts and techniques that form the basis for the presented works in the following chapters of this thesis.

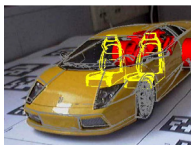
AR is an interdisciplinary field that involves, besides many technical aspects, also human computer interaction, perception, psychology or physiology. While the present works are mostly limited to the technical parts, encompassing computer graphics and computer vision, this dissertation specializes on the visually coherent presentation of virtual objects in AR environments and is therefore a topic in the field of computer graphics.

Focusing on the presentation is possible due to the extensive research, mainly driven by the computer vision and robotics community, that continuously provides more and more sophisticated localization and tracking techniques, enabling the registration of virtual content within the real world. Closely coupled to vision-based localization is the reconstruction of the real-world environment based on geometrical features. This chapter provides an overview of the concepts and techniques, that the thesis relies on. It is based on the book of Schmalstieg and Höllerer [SH16], which I refer to, for more in depth details.

Even when dealing with the presentation of virtual content in AR, the scope of this thesis needs to be defined more precisely. As JACOBS and LOSCOS state in their report [JL06]: Not all AR techniques are designed to make users believe that virtual objects seamlessly blend into the real world. Sometimes, it is desired to provide additional context information [KMS07], maybe by simple annotations [Ros+95]. In other cases, we might want to emphasize certain real elements instead of adding completely new content. Even stylized or non-photorealistic rendering can be relevant for some use cases [FBS05]. The methods presented in this thesis however, aim to create such seamless and coherent augmentations.

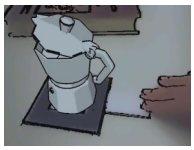
To combine virtual and real objects such that they perfectly blend up to the point where a user cannot distinguish virtual from real elements, three major topics need to be dealt with [SH16]:

GEOMETRIC REGISTRATION This is the foundation of coherent rendering. It mainly relates to camera parameters but also involves knowledge about the geometry of real-world surfaces. Extensive research has been conducted to determine the *Extrinsic Camera Parameters*, i.e., the position and orientation of the camera in the real-world 3D space. While this *camera pose* usually changes during interactions in the scene, there is a set of camera parameters that stay constant, namely the *Intrinsic Camera Parameters*, which include focal length, focal center, aspect ratio and usually parameters of modeled lens distortions. With intrinsics and extrinsics given, we are already able to provide important visual cues, that allow the user to perceive the location of virtual objects in space. That means, we are able to display virtual objects with correct size, orientation and perspective. If users know about the size of the virtual object, they can estimate how far away the object must be. For a known distance, they can estimate how big an object is compared to present real-world objects. However, these cues are very basic and allow only for very coarse estimates. With knowledge about the scene geometry we are able to consider occlusions, too. Objects in the front occlude objects that are farther behind, regardless of whether they are virtual or real. From our experience in the real world, we also expect that objects cast shadows onto each other. Shadows can help to resolve the discrepancy between size and distance as shown in Figure 3.1. When the geometry of real surfaces is known, we can visualize shadows that provide another important visual cue and help the user to assess the position of an object [WFG92, Hu+00], even when the correct light condition in the scene is unknown. Besides improving the shadow quality by considering the real-world light sources, there are further cues to support the user, e.g., atmospheric attenuation for very distant objects or showing fine details on the surface, that are only visible when observed from a close position and that help to understand the shape of the object. An important cue for this thesis is the shading of the object, but this is part of the next aspect of visual coherence.

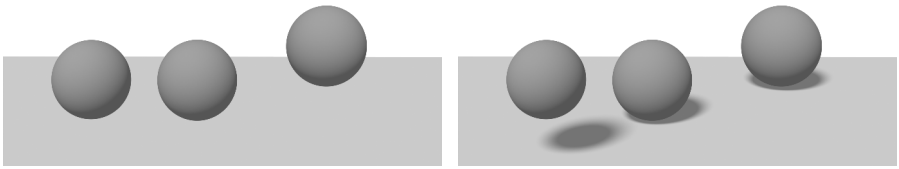


Courtesy of KALKOFEN *et al.* [KMS07]

PHOTOMETRIC REGISTRATION This aspect deals with the interaction of light between virtual and real objects. By applying light transport simulations or corresponding approximations, the virtual objects are shaded with respect to the current light conditions in the real-world environment. This aspect and the previous one are most important when trying to make virtual objects blend



Courtesy of FISCHER *et al.* [FBS05]



(a) No Shadows (b) With Shadows

Figure 3.1: Importance of Shadows
Shadows provide an important visual cue that is required to assess the relative position and size of objects.

perfectly into the real world. Because of the imperfect nature of acquisition and display devices, further processing steps can be required to eventually convince the users (see the last aspect). However, to achieve a correct or at least plausible appearance in terms of light transport, certain knowledge about the real environment is required. Depending on the desired level of quality and the type of light paths between the virtual and the real world, that are supported by the approach, a different amount of knowledge about the scene is needed. While simple approaches use coarse approximations of the direction and intensity of environment light, sometimes even manually defined by the user. More sophisticated approaches rely on inverse light transport simulation to recover parameters from images. A commonly used classification of illumination methods was presented by Jacobs and Loscos [JL06]:

- *Local Common Illumination*
  Geometry and radiance information from the environment are used to illuminate virtual objects. Correct occlusion handling and shadows between real and virtual objects are also considered features of this class, although they are technically non-local effects.
- *Global Common Illumination*
  While local common illumination considers only direct light, the global variant takes secondary light and higher order bounces into account. It is called *global*, because to achieve these kind of effects, global illumination simulation techniques are required.
- *Relighting*
  Techniques that focus on changing the real surface shading by virtually altering the illumination or the material of the real object. These methods usually require a sophisticated knowledge of the real illumination condition and the material properties of real surfaces. Note, that the changing of local surface properties also has a global impact on the entire scene.
- *Inverse Illumination*
  Applying global illumination techniques to estimate the real illumination condition and material properties as precise as possible from input images of a scene. Eventually, this allows to exactly reproduce the content of the input footage by rendering the reconstructed data.

Chapter 4 provides an extended overview of methods that are addressing the issue of photometric registration as they form the largest part of the related work of this thesis.

CAMERA SIMULATION This last aspect accounts for the physical behavior of the camera, the sensor and the lens. Simulating the behavior of these components is important while acquiring environment information and for displaying virtual objects as well. During the acquisition, we try to reason about the RGB input image to deduce measurements of physical quantities, i.e., radiance in most cases. Therefore, knowledge about the internal processes of the cameras is required. Especially low-cost consumer products like the cameras of mobile devices exhibit various kinds of distortions, artifacts and even unspecified software manipulations that influence the image. Using more professional equipment, like DSLR cameras, we have detailed control over the camera parameters. Without further knowledge about the actual internal processes, it is possible to fix settings, like the white balance and calibrate the camera by capturing several images with different exposure – a quantity that depends on the aperture, shutter speed and ISO sensitivity – by solving for the *Camera Response Curve*. Given the response curve, we can deduce linear radiance on real-world surfaces from Low Dynamical Range (LDR) color values, stored per pixel in the input image. Besides the response of the camera system, there are several other features that can be accounted for. As PAUL DEBEVEC *et al.* stated: *"the patterns of film grain* [...] *should match"* [Deb98], but various additional effects like blur, vignetting, chromatic aberrations or other artifacts, e.g., introduced by the Bayer mask, can be accounted for to improve the quality during the acquisition. The same holds for displaying the augmented image. The level of realism can be increased by applying the same effects to the augmented elements.

Assuming we are able to perfectly reproduce all of these effects and achieve a seamless blending, we can go one step further and also account for the properties of the display device. Depending on the modality, different effects are introduced, that can be accounted for in the ideal case. For a *Hand-held Video See-Through Device* (see Section 3.5) for instance, a calibration of the display can eventually allow to realize an AR experience in the sense of *Magic Lenses* [Bie+94, Vie+96], where the frame of the device acts as a window into the augmented world and the appearance of real-world objects on the screen are equal to the appearance next to the device[7]. In the following sections, some of the individual topics are discussed in more detail.

3.2 INTRINSIC PARAMETERS

Since a solid intrinsic calibration is required for all other tasks in AR, it is the first topic we need to address. As stated earlier, intrinsic parameters (or simply *Intrinsics*) contain fixed parameters like focal length or aspect ratio, that describe the projection of the 3D scene into 2D image space. They are highly important because *"otherwise the object may seem too foreshortened or skewed relative to the rest of the picture"* [Deb98]. The set of parameters however, depends on the camera or lens model we are using. For real-time rendering, the focus is generally on rather simple models, i.e., *pinhole cameras* and *fish-eye cameras* throughout this thesis. In the following both models are discussed, including calibration techniques that have been used to realize the systems presented in Chapters 5 to 7. For more advanced camera models, I refer to the PBRT book [PJH16].

7 There are some additional requirements, i.e., head tracking for a correct perspective, camera input with a sufficiently large field of view to compensate changes in perspective and ideally an autostereoscopic presentation.

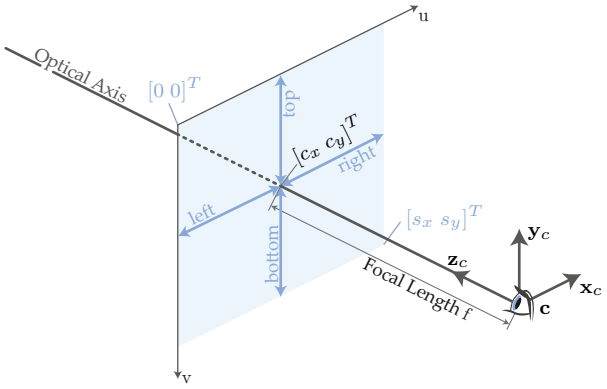


Figure 3.2: Pinhole Camera in Computer Graphics

3.2.1 *The Pinhole Camera*

This model is simple and known for a long time as *Camera Obscura*. Actually, the first known records are from Mo Ti, a Chinese philosopher of the 5^{th} century B.C. [Dav99]. Da Vinci also described the idea in his notebooks[8]:

> *I say that if the front of a building – or any open piazza or field – which is illuminated by the sun has a dwelling opposite to it, and if, in the front which does not face that sun, you make a small round hole, all the illuminated objects will project their images through that hole and be visible inside the dwelling on the opposite wall which may be made white; and there, in fact, they will be upside down* [...]
>
> — Leonardo Da Vinci [Ric70]

In computer graphics, we apply a trick to further simplify computations, which is not possible in the real world. We define an image plane outside the room, in front of the pinhole and virtually measure the radiance of light rays that would pass through the plane and the hole. Then, the position of the pinhole is interpreted as the position of the camera – or the eye. Figure 3.2 illustrates this along with further parameters that allow to model properties of real cameras. The position of the camera $\mathbf{c}$ defines the origin of the camera space, where $\mathbf{z}_c$, the optical axis, points towards the viewing direction, $\mathbf{x}_c$ to the right and $\mathbf{y}_c$ upwards, i.e., we use a left-handed system. The image plane of size $\begin{bmatrix} s_x & s_y \end{bmatrix}^T$, measured in pixels, is aligned perpendicular to $\mathbf{z}_c$, where the distance to the camera position $\mathbf{c}$ is called focal length f, also measured in pixels. In the illustration, it is assumed that the pixels of the image are squares. If that is not the case, we compute $f_x = F/p_x$ and $f_y = F/p_y$, where F is the focal length in world units and $\begin{bmatrix} p_x & p_y \end{bmatrix}^T$ the size of a pixel in world units. It is also possible that the optical axis does not match exactly with the center of the image – imagine that the lens of your camera is shifted a little bit and not perfectly aligned to center of the sensor. Therefore, we define the principle point $\begin{bmatrix} c_x & c_y \end{bmatrix}^T$, an in-plane translation of the image that is also defined in pixels. The origin of the image coordinate system is usually defined as the top

8 Actually, Da Vinci extends this thought experiment to holes at multiple points in the wall and than to all points in time and space. Hence, he actually describes the *Plenoptic Function* (see Section 4.1.1), which is why Adelson and Bergen also use this quote to motivate their work [AB91].

Figure 3.3: Intrinsic Camera Calibration Result
An camera image augmented by two cubuids. The green one is rendered after intrinsic calibration, while the red one is rendered with guessed parameters. A typical checkerboard, that is used for the calibration, is visible in the background. Parameters specified by the manufacturers are often limited to an average field of view parameter and usually achieve results somewhere in between. However, when developing for multiple platforms and a various different devices, the parameters given by the manufacturer are often not a practical solution. Image courtesy of Clemens Arth *et al.* and AR4.io.

left corner of the image. Hence, $\begin{bmatrix} c_x & c_y \end{bmatrix}^T$ is usually about $\begin{bmatrix} s_x/2 & s_y/2 \end{bmatrix}^T$. Some arbitrary point $\mathbf{x} = \begin{bmatrix} \mathbf{x}_x & \mathbf{x}_y & \mathbf{x}_z \end{bmatrix}^T$, given in the camera coordinate system, can be projected to an image space position $\begin{bmatrix} \mathbf{x}_u & \mathbf{x}_v \end{bmatrix}^T$ by the following Equations:

$$\mathbf{x}_u = f_x \ {}^{\mathbf{x}_x}/_{\mathbf{x}_z} + c_x$$
$$\mathbf{x}_v = f_y \ {}^{-\mathbf{x}_y}/_{\mathbf{x}_z} + c_y.$$

For rendering on the GPU, this projection is usually realized by a linear matrix multiplication in homogeneous space followed by a perspective division. Therefore, the graphics Application Programming Interfaces (APIs) provide methods to setup the projection matrix using the visualized offsets: *left*, *right*, *bottom* and *top*. Note, that the offsets in Figure 3.2 correspond to a *Near Plane Distance* that equals the focal length. The APIs also allow to specify the projection matrix using the aspect ratio, ${}^{s_x}/_{s_y}$, and a *Field of View* parameter α, which is computed as:

$$\alpha_y = 2.0 \arctan \frac{0.5\, s_y}{f_y}.$$

for a vertical field of view. Note, that α spans the entire field of view, while it is also common to specify the angle between the optical axis and the (upper) frustum border and thereby half the field of view. For more details, I refer to any text book or lecture on computer graphics as well as to the documentation of the chosen API. The most widely used algorithms for intrinsic camera calibration, e.g., the one by Zhang [Zha00], additionally estimate a skew parameter to compensate for non-perpendicular image axes. Additionally, they provide coefficients to compensate radial and tangential lens distortions, which are only required for real lenses, as virtual pinhole cameras do not have lenses.

For our research, we use the implementation provided by OpenCV[9] or the Vizario.cam[10] app on the android platform. In both cases a set of images,

9 OpenCV. *Open Source Computer Vision Library, Version 3.3.0.* Documentation, 2017.
10 AR4.io. *Vizario.cam.* Android App, 2016.

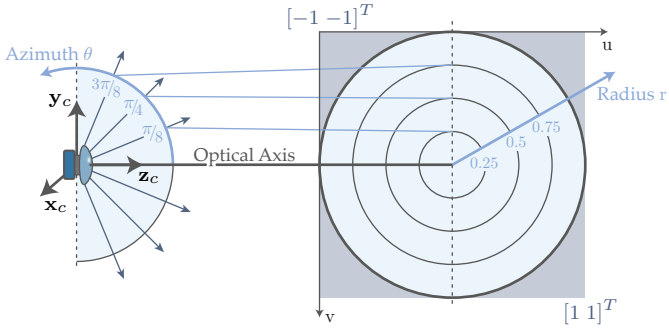


Figure 3.4: Azimuthal Equidistant Projection

showing a checker pattern of known size that has been captured from multiple angles, is required as input. After extracting the chessboard corners in all images, the tools run a global optimization, that iteratively solves for the intrinsic parameters, reconstructs the corner positions in camera space, projects them into image space using the current set of parameters, and minimizes the projection error. Figure 3.3 emphasizes the importance of such an intrinsic calibration.

Important to note is that the calibration can depend on the settings of the camera and the lens. This means that resulting parameters are valid only for one specific focal length (in case of zoom-lenses) and for the selected aspect ratio as well as resolution.

3.2.2 *The Fish-eye Camera*

The maximum field of view that can be achieved with pinhole cameras is limited. In theory, a projection can be defined as long as the field of view is below 180° where the focal length would be zero. Real-world rectilinear lenses however, even with a full frame sensor, are assumed to have the smallest practical focal length of about 15 mm and even more for smaller sensors [Joh03]. A common solution to achieve larger fields of view with a single image is by using a fish-eye lens. Their mapping of arbitrary points of the scene into the 2D image space is different from the pinhole model. Instead of using a perspective projection, we assume a hemisphere at the position of the camera $\mathbf{c}$. The hemisphere points towards the viewing direction $\mathbf{z}_c$, which is again referred to as the optical axis. Analogous to the pinhole camera, we define $\mathbf{y}_c$ as upwards and $\mathbf{x}_c$ as right direction. Each point in camera space can now be expressed in spherical coordinates $\left[\theta\ \varphi\ \rho\right]^T$, where θ is defined as the angle between $\mathbf{z}_c$ and the direction of the considered point. φ defines the angle of rotation around the optical axis and ρ is the distance between the point and the camera position $\mathbf{c}$. The image produced by the fish-eye projection has the shape of a disk, where the color captured for the direction of the optical axis is mapped to the image center. The angle φ is not changed and specifies the direction within the disk with respect to the center, whereas the azimuth angle θ is transformed by some function $f(\theta)$ that maps θ to a radius r. Figure 3.4 illustrates that model, while assuming an image space that is defined as $[-1, 1]^2$ and a function f that maps θ linear onto the radius r. This ideal

equidistant projection of an arbitrary point $\mathbf{x}$ in camera space is computed in two steps. First, $\mathbf{x}$ is transformed into spherical coordinates:

$$\begin{aligned} \mathbf{x}_\theta &= \arccos \frac{\mathbf{x}_z}{\rho} \\ \mathbf{x}_\varphi &= \arctan 2 \frac{\mathbf{x}_x}{\mathbf{x}_y} \\ \mathbf{x}_\rho &= \|\mathbf{x} - \mathbf{c}\|_2 . \end{aligned} \tag{3.1}$$

In the second step, the mapping function f is applied and the image space coordinates are computed. Note, that ρ is not used, but it can be used as depth in a z-Buffer:

$$\begin{aligned} r = f(\mathbf{x}_\theta) &= \frac{2\,\mathbf{x}_\theta}{\pi} \\ \mathbf{x}_u &= r \cos \mathbf{x}_\varphi \\ \mathbf{x}_v &= -r \sin \mathbf{x}_\varphi . \end{aligned} \tag{3.2}$$

To calibrate a real-world fish-eye lens, we use the OCamCalib Toolbox[11] by Davide Scaramuzza. One goal of the calibration is to approximate the mapping function f of the real-world camera. In this case, f is modeled as a polynomial. As suggested by the author, we use a 4^{th} order polynomial and therefore estimated five coefficients. Similar to the case of the pinhole camera, the optical axis can be off-center and not perfectly perpendicular to the image plane. Additionally, the pixel may not be squared. To account for these issues, the toolbox also solves for affine transformations to model the effects. For additional information concerning the method, I refer to the corresponding papers [SMS06a, SMS06b, RSS08].

For easier evaluation during run-time, we created a lookup table based on the calibration result. This lookup table can be stored in a texture and is used to re-sample the camera input into an image with perfect equidistant projection as shown in Figure 3.4. Equation (3.1) and (3.2) and their inverse functions allow for easy conversion between the 2D image coordinates and the 3D view space, assuming the depth ρ was stored, too. By altering the mapping function f in Equation (3.2), we can support field of views different from 180° ($\alpha = \pi$):

$$f(\theta) = \frac{2\,\theta}{\alpha},$$

which allows up to 360° but also supports smaller field of views without wasting additional texture space. In this case, α is the only parameter that is stored along with the lookup table. Together, they can be seen as intrinsic parameters of the fish-eye camera.

3.3 Extrinsic Parameters

In the previous section we worked in the local coordinate system of the camera. This system is also called *Camera Space*, *Eye Space*, *View Space* or *View Reference Space*, which all refer to the 3D coordinate system with the origin at the position of the camera and the z-axis aligned with the view direction. The position of objects in the scene are typically defined in global coordinate system referred to as *world space*. The origin of this system is fixed and coordinates of static objects are constant, whereas in view space, the coordinates

11 Davide Scaramuzza. *OCamCalib: Omnidirectional Camera Calibration Toolbox for Matlab.* Project Website, 2013.

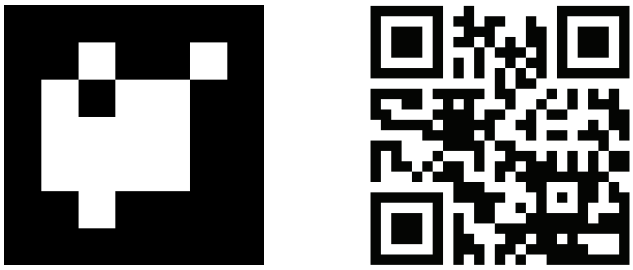
of static objects change when the camera is moved. Both spaces are related by the *Camera Pose*, the position and orientation of the camera in world space, which are also referred to as extrinsic parameters or simply *Extrinsics*. The pose can be expressed as a linear transformation consisting of a translation and rotation, the *View Transformation*. Since this transformation is invertible, it allows to transform positions between world and view space. Together with the projection defined by the intrinsic parameters, we are able to compute the exact location in image space for any surface point given in world space. For a given position in image space, we can at least specify a world space ray, starting at the camera position, on which the surface position visible in the image lies. With knowledge about the depth, i.e., the distance between the visible point and the camera[12], the 3D world space position can be reconstructed exactly.

In the context of AR, the process of estimating the six degrees of freedom for a pose of an object in three-dimensional space is often referred to as *Tracking*. In general arbitrary real-world objects can be tracked, but the camera pose is one that is required in any AR scenario. The following subsections provide an overview of the basic optical methods that are used in the later chapters of this thesis. The ones described below and various other techniques can be found in the book of Schmalstieg and Höllerer [SH16].

INSIDE-OUT VS. OUTSIDE-IN TRACKING There are two different types of tracking strategies. Inside-out tracking relies on sensors attached to the tracked objects, e.g., the camera or gyroscope, compass and accelerometer. The tracked object is actually the active component. The sensors are used to estimate the own pose by observing stationary, mostly passive, features in the environment, for instance special markers, corners of objects, details in the texture of objects, or in case of the gyroscope, the direction of gravity. Inside-out tracking often provides good estimates of the orientation, since the observed features are usually farther away, compared to the size of a handheld device. This is true, especially for computer vision-based approaches. The estimated position however, often lacks of precision because the estimates are often based on some sort of integration, where errors are propagated over time. In general, inside-out techniques are often preferred as they require no – or at least only minor – preparation of the environment.

The alternative is to use outside-in methods, where the sensors are placed in the environment and the tracked objects are passive. This allows to estimate the position of arbitrary (passive) objects but requires the placement of sensors in the environment, which usually limits the space of interaction. Here, the sensors observe features of the dynamic object. The measurements of multiple, calibrated sensors allows to precisely estimate the position of the tracked object. The precision of the estimated orientation is low compared to inside-out tracking, while the precision gets worse the smaller the object becomes. Especially, when tracking a camera, this can be drastically, where a low error of 1° in the orientation already leads to an offset of about 0.5 cm when augmenting an object at a distance of 1 m. However, since any object can be tracked that is not necessarily visible to the display device, potentially different scenarios can be supported.

12 The depth of a point in image space can be defined differently, e.g., as the Euclidean distance to the camera or the distance of the point projected onto the optical axis. The distances can for instance be given in world space units, normalized world space units between near and far plane or logarithmically scaled as a result of the perspective projection using the rasterization pipeline.



(a) ArUco (b) QR Code

Figure 3.5: Binary Square Markers
Various different marker designs have been proposed. A popular example are the ArUco marker by Sergio Garrido-Jurado *et al.* [Gar+14], which is also available in OpenCV. QR codes are more complex but allow to encode more information than a simple ID.

3.3.1 *Marker Tracking*

DETECTING FLAT 2D MARKERS A group of simple but quite robust tracking techniques are marker-based methods. Here, a known and easily detectable object is placed in the environment. The checkerboard in Figure 3.3 is one example of such a known object, but smaller square markers, like the ones shown in Figure 3.5, are more popular as the encoded IDs allow to distinguish different markers used in the same scenario. Since the marker is known, it can be classified as model-based technique. To be able to estimate a camera pose from such 2D binary marker, visible in an input image, a proper intrinsic calibration is required.

The basic steps for detecting such markers are the following: The input image is converted into a binary black and white image, usually by applying a threshold after some contrast enhancement steps. This is followed by the search for line segments and closed loops, which is the most expensive part of the algorithm. Each quadrilateral loop of sufficient size is a marker candidate. Sampling at a predefined set of points inside the quadrilateral shape reveals the ID as well as the top, left, right and bottom edge of the potential marker. If the sampling result does not match the pattern of a valid marker, the candidate is rejected. Otherwise, the pose is estimated by mapping the detected planar shape to a square of specified size using a homography. The resulting pose can be interpreted as the position of the camera in the local coordinate system of the marker. Inverting the transformation yields the pose of the marker in view space. An additionally provided world space pose of the marker allows to convert between the involved spaces. A detailed description, that also consider the intrinsic parameters of the camera, can be found in any computer vision text book, e.g., the ones of Szeliski [Sze11] or Hartley and Zisserman [HZ04].

In Chapter 5, stationary cameras with fish-eye lenses are used to capture the environment. To estimate their extrinsic parameters, we use a checkerboard with a known pose in world space. Figure 3.6 is used to illustrate the pose estimation in 2D. The problem could also be solved by finding a homography, but starting with the fish-eye calibration that offers an equidistant mapping, we can show a slightly different approach by setting up a different linear system:

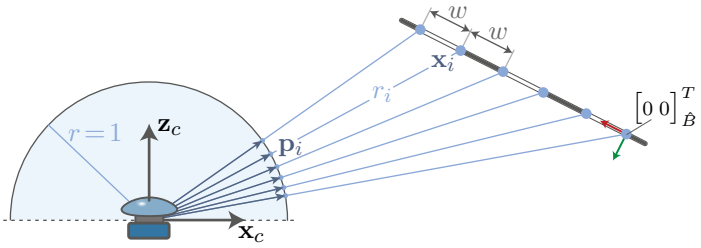


Figure 3.6: Un-project a Grid into View Space

Based on Equation (3.1) and (3.2) we can reconstruct the direction of detected checkerboard corners in view space. Assuming a radius of 1, these corners are located on a unit sphere around origin and denoted as points $\mathbf{p}_i$. We know, that corners actually lie somewhere along the rays on a planar grid of unknown pose but known size. Assuming the checkerboard was in the upright position, we can specify $\mathbf{x}_i$, corresponding to $\mathbf{p}_i$, as the point located at an unknown radius r_i on the checker board. We also know, that the checkerboard is a uniform grid and that points along one direction have a known distance of w.

For solving the problem in 2D, we can setup the following equations: From the second observation, we can conclude that inner points have two neighbors at the same distance in opposite directions and thereby:

$$\begin{aligned} & \mathbf{x}_{i-1} - \mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i+1} \\ \Rightarrow \qquad & -\mathbf{x}_{i-1} + 2\mathbf{x}_i - \mathbf{x}_{i+1} = 0. \end{aligned}$$

For n points, this leads us to $n-2$ such relationships, hence $2(n-2)$ equations for the linear system in the 2D case of the example because of the two components.

The other observation leads to more obvious equations as:

$$\begin{aligned} & r_i \mathbf{p}_i = \mathbf{x}_i \\ \Rightarrow \qquad & -\mathbf{x}_i + r_i \mathbf{p}_i = 0. \end{aligned}$$

However, we need to avoid the trivial solution by adding another constraint, which can be achieved by fixing the scale of one point, e.g., by moving one $\mathbf{p}_1$ to the right-hand side:

$$-\mathbf{x}_1 = -\mathbf{p}_1.$$

This leads to one relationship for each point (where one is special) and $2n$ equations for the block-wise diagonal linear system with unknown $\mathbf{x}_i$ and r_i shown below. Solving the system gives the coordinates $\hat{\mathbf{x}}_i = \begin{bmatrix}\hat{\mathbf{x}}_{i_x} & \hat{\mathbf{x}}_{i_x}\end{bmatrix}^T$ up to a scale factor s, which is caused by fixing one point. The factor in turn, can be retrieved by considering the average distance of neighboring points, or just the distance of the first two points, scaled by the known distance w:

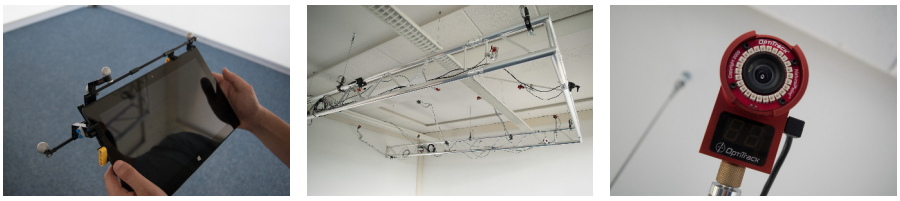
$$\begin{aligned} s &= \frac{w}{\|\hat{\mathbf{x}}_2 - \hat{\mathbf{x}}_1\|_2} \\ \mathbf{x}_i &= s\,\hat{\mathbf{x}}_i. \end{aligned}$$

$$
\left[\begin{array}{ccccccc|c}
-1 & 0 & 2 & 0 & -1 & & & \\
 & -1 & 0 & 2 & 0 & -1 & & \\
 & & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\
 & & & -1 & 0 & 2 & 0 & -1 \\
\hline
-1 & & & & & & & 0 \\
 & -1 & & & & & & 0 \\
 & & \ddots & & & & & \mathbf{p}_{2_x} \\
 & & & \ddots & & & & \mathbf{p}_{2_y} \\
 & & & & \ddots & & & \ddots \\
 & & & & & \ddots & & \ddots \\
 & & & & & & -1 & \mathbf{p}_{n_y} \\
 & & & & & & & -1 \quad \mathbf{p}_{n_y}
\end{array}\right]
\left[\begin{array}{c}
\hat{\mathbf{x}}_{1_x} \\ \hat{\mathbf{x}}_{1_y} \\ \hat{\mathbf{x}}_{2_x} \\ \hat{\mathbf{x}}_{2_y} \\ \vdots \\ \hat{\mathbf{x}}_{n_x} \\ \hat{\mathbf{x}}_{n_y} \\ \hline r_1 \\ r_2 \\ \vdots \\ r_n
\end{array}\right]
=
\left[\begin{array}{c}
0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ \hline \mathbf{p}_{1_x} \\ \mathbf{p}_{1_y} \\ 0 \\ \vdots \\ 0
\end{array}\right]
$$

The 3D case is very similar, except that there are three equations for each relationship. Additional equations to ensure the distance of grid points in the second dimension can be defined analogously. Eventually, $\mathbf{x}_1$, the principle directions of the un-projected grid and their cross-product define the basis of the checkerboards local coordinate system and also the pose of the camera in checkerboard coordinates. With the help of the world pose of the board, we can finally compute the world position of the fish-eye camera. The same method could be used to estimate the pose of cameras with rectilinear lenses, too, as the image space positions can be projected onto a unit sphere in view space.

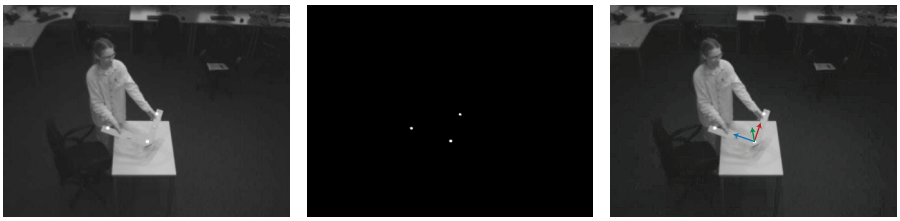
DETECTING SPHERICAL MARKERS These passive markers are used for outside-in tracking, e.g., in the context of motion tracking in movie and game productions. Multiple of those small light-weight plastic spheres, coated with highly retro-reflected film, are composed to tracking targets, where each target consists of three or more markers in a unique rigid constellation. Figure 3.7a shows an example tracking target we use for the approach in Chapter 5.

The coating of the markers works like retro-reflected foil known from safety clothing or traffic signs. A large amount of the incident light is reflected back towards the sender. This property is utilized by special cameras that are equipped with infrared emitters, as shown in Figure 3.7c. The emitted light



(a) Tablet with Markers (b) Tracking System (c) Infrared Tracking Camera

Figure 3.7: Infrared Tracking Setup
A common tablet, instrumented with spherical markers (a), using an infrared outside-in tracking system (b). The system consists of special cameras (c) equipped with a circle of infrared LEDs.



(a) Image of IR Camera (b) Marker Candidates (c) Estimated Pose

Figure 3.8: Infrared Tracking Example
The image (a) is produced by one of the tracking cameras visible in Figure 3.7. After applying a threshold, bright spots are considered marker candidates (b). After associating a predefined marker constellation, referred to as tracking target, the pose of the tracked object can be estimated (c).

is not visible to humans but because of the retro-reflection, markers appear as very bright spots in the camera image. Even without a special filter, markers can be detected easily after applying a threshold (see Figure 3.8b). A blob detection is conducted on the resulting binary image to identify marker candidates, that fulfill certain image space size constraints. In the next step, epipolar geometry is used to establish correspondences in two or more camera images. Therefore, the camera system needs to be calibrated properly before starting the tracking, but as long as the cameras are not moved, this is done only once. After finding correspondences, the 3D position of the marker is triangulated by identifying the point, at which the world space rays from each camera through the corresponding marker in the image get closest, as the rays do not intersect exactly. The extracted 3D positions of the marker candidates are then associated to the tracking targets. Even though the target constellation is known, the association can become a complex task, especially, when many markers are visible or many different constellations are expected. It is possible that some markers are hidden from the cameras because of user interactions, or the triangulation is not successful because the marker is only visible in a single camera image. Hence, all permutations of candidates have to be tested against the known constellations or subsets of them. Thus, the robustness of the system is improved by using more than the minimum number of markers per target. Eventually, when a match was found, the pose of the object is defined by the affine rigid transformation of the target, which usually also has a local coordinate system, as shown in Figure 3.8c. Such a system is used during the experiments in Chapter 5 for estimating the position of the tablet, moving real-world paper boxes, and to determine the position of the checkerboard when estimating the world-space positions of the fish-eye cameras.

To be able to estimate the extrinsic parameters for the back-facing camera of a hand-held device using an outside-in tracking system, we also need to account for the rigid transformation between the local coordinate system of the tracking target and the actual position of the (virtual) camera. Therefore, we use another calibration step to estimate this constant transformation. This involves the tracking of a checkerboard. Small patches of retro-reflective foil are placed on the edges of the board to allow for infrared tracking. At the same time, the device, instrumented with spherical markers, is also tracked by the infrared system. Now, one or more images of the checkerboard are captured. Performing a vision-based pose estimation using the visible checkerboard gives the pose of the camera in the local coordinate system of the checkerboard. Such

a pose is specified by a rotation and a translation $[R|t]$. Representing this pose as 4×4 matrix and inverting the result gives the view transformation $\hat{V}$, which transforms homogeneous coordinates from the local space of the checkerboard $\mathbb{R}^3_{\hat{B}}$ into view space $\mathbb{R}^3_{VS}$:

$$\hat{V} : \mathbb{R}^3_{\hat{B}} \rightarrow \mathbb{R}^3_{VS}.$$

Additionally the poses provided by the infrared tracking system are considered, i.e., the world space pose of checkerboard $[R|t]_{\hat{B}}$ and the world space pose of the device $[R|t]_{\hat{D}}$. Writing these poses as 4×4 matrices (not inverted) gives the mappings from the corresponding local space into world space $\mathbb{R}^3_{WS}$:

$$\hat{B} : \mathbb{R}^3_{\hat{B}} \rightarrow \mathbb{R}^3_{WS}$$
$$\hat{D} : \mathbb{R}^3_{\hat{D}} \rightarrow \mathbb{R}^3_{WS}.$$

The calibration matrix C we are interested in, maps from local device coordinates into the view reference space. Therefore, we concatenate the available transformations as follows, assuming that we multiply the vector to transform from the right, and stored the matrix for later usage:

$$C : \mathbb{R}^3_{\hat{D}} \rightarrow \mathbb{R}^3_{VS}$$
$$C = \hat{V}\, \hat{B}^{-1}\, \hat{D}.$$

During interactive scenarios, the checkerboard is not needed anymore. The infrared tracking system continuously provides world space poses of the device $[R|t]_D$ and thereby a transformation, D from the local device space into world space. The view matrix V that is used for rendering computes as:

$$V : \mathbb{R}^3_{WS} \rightarrow \mathbb{R}^3_{VS}$$
$$V = C\, D^{-1}$$
$$\mathbf{x}' = V\, \mathbf{x}.$$

It allows to transform world space positions $\mathbf{x}$ into the local space of the camera $\mathbf{x}$', which is usually followed by a projection into image space.

DETECTING VS. TRACKING As the term *tracking* already suggests, it means following one or multiple objects of interest in the scene and to continuously determine their location. In 3D tracking, this location is the six degrees of freedom pose discussed above. By incorporating assumptions about the dynamics of tracked objects and thereby predicting future positions based on observations from the past, it is possible to estimate the pose of an object, even if the target is currently not visible or partially hidden.

In contrast, the pose estimation by detection (tracking by detection) is not based on a prediction model and temporal consistency is not addressed directly. However, detection is a required step while tracking an object, which can be improved by knowledge of the dynamic model. The prediction step allows to limit the search area for markers or features and thereby reduces the computational effort. This is also true for associating spherical marker candidates to the tracking targets in case of the infrared tracking discussed before. In case the prediction-based tracking fails, a full detection step can be used as backup until the target is found again.

(a) FAST (b) SURF

Figure 3.9: Extracted Natural Features

Features detected by two common feature detectors. Image (a) shows the result of the FAST detector acquired within 6.7 ms, while the features in the other image (b) have been found within 158 ms using a SURF detector. The implementation of the MATLAB Computer Vision System Toolbox was used on an INTEL CORE I7-6700K for the shown image with approximately 2.5 megapixels. Image courtesy of SAALE-UNSTRUT-TOURISMUS E.V. UND MITGLIEDER.

3.3.2 *Natural Feature Tracking*

Instrumenting the environment or the devices used with markers is no suitable for all scenarios. Outside-in tracking is further limited to the area that is covered by the tracking cameras and the system itself is only suited for special environments, like laboratories, but not for usage in a living room, due to their size and price. Hence, non-invasive methods without any additional hardware or markers that can be used for tracking in larger areas are desirable. However, they are more complex and computationally more expensive.

In recent years, techniques based on *Natural Features* have become more and more popular, as they fulfill these requirements. As we apply this kind of tracking only in form of a service that is provided by the mobile device used in Chapter 7, this section will only convey the concepts based on the elaborations of SCHMALSTIEG and HÖLLERER [SH16].

Natural features, or *interest points*, are visual properties that are already present in the real world. Many different features can be used for tracking, as long as they are easy to find in an image, their position in the environment must be independent from the vantage point, and they should be robust against changes in illumination. Corners, edges or features in texture of the objects are often the fundamental elements that constitute such a feature. It is important to realize, that one feature alone – in contrast to a marker – does usually not allow to reconstruct the camera pose. Because many of the interest points appear repeatedly in an image, e.g., around windows in the facade of a building, they are not unique. Hence, the constellation of multiple interest points needs to be considered.

Two examples for extracted features are shown in Figure 3.9, using two common *Feature Detectors*. Various detectors and variations have been proposed, which differ in terms of quality, speed and robustness. In the examples, FAST, originally proposed by ROSTEN and DRUMMOND [RD06], and SURF, proposed by BAY *et al.* [Bay+08], have been applied. After detecting features, they are stored in a data structure optimized for matching, which is referred to as *Feature Descriptor*. The fast and reliable matching of descriptors found in two

images taken from different vantage points or matching the descriptors of an acquired image against a previously captured model, is a key element of natural feature tracking. This matching involves a search or approximated search for corresponding features. A simple criteria for a good match is the Euclidean distance between descriptors represented as feature vector. The closer the distance, the better the match. Incorporating the handling of outliers makes the next steps easier, in which the camera pose is estimated based on the matching. Therefore, the so called *Perspective-n-Point* (PnP) problem is solved to reconstruct the pose based on n correspondences between 2D and 3D points. The latter are the matched features of the model, that are given with 3D world coordinates. The outlier handling can also be realized by repeatedly solving the P3P on randomly chosen matches. By applying the transformation to the remaining features and computing the distance between the corresponding feature positions, outliers can be detected. Eventually, only inliers are used to estimate the final camera pose. This method is known as RANSAC introduced by FISCHLER and BOLLES [FB81].

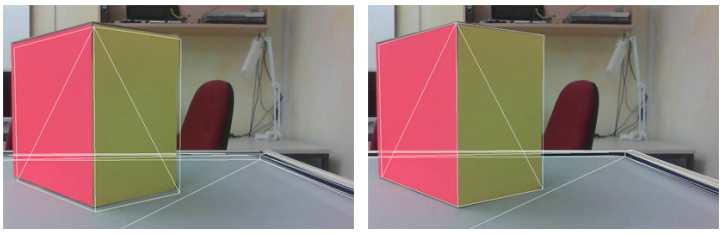
The steps described above, estimate the camera pose based only on the input of the current image. The method is thereby an example of tracking by detection. Because of the number of features, incremental approaches that use information of previous steps in time can help to meet the performance goals. To realize the prediction of future feature positions based on a motion model, it is common to consider the *Optical Flow*, which is a vector field that describes the movement of points in a sequence of consecutive images (see *Kanade-Lucas-Tomasi Tracking* [BM04]).

So far, we assumed a given feature set of the environment, the model. Tracking methods that start without a predefined model, also use models for tracking, but these are created on-the-fly. In *Visual Odometry* [NNB04], the camera pose is updated incrementally on a frame-to-frame basis by estimating the essential matrix that relates two consecutive frames. However, if the features that are used to compute transformation are not visible in the image anymore, they cannot be identified when reappearing in one of the following steps. Storing the detected features in a spatial data structure that allows for matching features over a longer period of time even when they were hidden, leads to Simultaneous Localization and Mapping (SLAM), a concept originally used in the field of robotics, just like visual odometry. One of the first real-time methods that used SLAM for recovering the trajectory of a camera was presented by DAVISON *et al.* [Dav+07]. They already showed an AR scenario that uses their *MonoSLAM* tracking.

While using simple incremental transformation updates, it is very likely that drifting errors are accumulated over time. Hence, when revisiting an area that has already been seen in the past, the position of features will probably show a gap. This issue is addressed by *Bundle Adjustment* [Tri+99] that allows to *close the loop* by minimizing the re-projection error of the features over time. For more details to the single topics and also for more advanced techniques, I again refer to book of SCHMALSTIEG and HÖLLERER [SH16].

The tracking service[13] provided by the mobile device used for the experiments in Chapter 7 applies several of these techniques in two separate phases. In the first phase, called *Area Learning*, the current position of the device, with respect to the starting position, is estimated by *Visual-Inertial Odometry*. As the name suggests, inertial measurements provided by gyroscope, compass and accelerometer are fused into the pose estimation based on visual

13 Google Developers. *Tango Documentation: Motion Tracking*. Project Website, 2016.



(a) Without Refinement (b) With Refinement

Figure 3.10: Refining Outside-in Tracking using Particle Filters
Results presented in the bachelor's thesis, that investigated a refinement strategy to improve the outside-in tracking, used for the approach in Chapter 5 and 6, by an 3D object tracking method. Image courtesy of Steve Wolligandt [Wol15].

features. Such a hybrid tracking was suggested by State *et al.* [Sta+96] for instance. During tracking, the system creates a feature map to match the current input against previously seen features. While learning the area, the device orients and positions itself within the previously learned area. The service also provides a drift correction for loop closure. Hence, the service relies on a SLAM-based tracking with bundle adjustment. When the area learning is completed, the map is used for the second phase or stored persistently for later sessions. In the second phase, the learned area does not change anymore. Instead, the pose is estimated by matching detected features to the feature map. This vision-based result is then fused with measurements of the inertial sensors. The inertial data is probably also used as motion model to select features or a spatial window for features to match.

3.3.3 3*D Object Tracking*

This subsection focuses on a different kind of model-based tracking. Here, the model is a given geometric representation of a real-world object. The goal is to find the camera pose, relative to the known object based geometric features – mostly edges and silhouettes – that are visible in the current camera image. Especially, if precise Computer-aided Design (CAD) data of the object to track is available, object tracking is often preferable as contours are usually well-defined even under changing light conditions and requires no special features in the texture of the object. Since edge detection is a rather simple image processing task, the computational cost for this first step is low. An artificial image of the model is rendered from the currently estimated camera pose. After extracting the edges from that rendering, distances between the edges of both images are computed along the edge normals. Then, a new camera pose, that minimizes these distances, is computed in an iterative manner [DC99]. Instead of using just one rendered edge image, the GPU can be used to evaluate multiple of them. Therefore, the particle filter approach of Isard and Blake [BI97, IB98] is used to create multiple camera pose candidates distributed in the area of the current estimation. Weighting the individual poses of the candidates based on their likelihood, which is derived from the distances between the edges, results the next estimated camera pose. New particles are then generated by considering a motion model. It is also possible to perform multiple iterations per frame using different particle densities to

refine the estimate. This 3D extension of the approach of Isard and Blake was presented by Klein and Murray [KM06].

We investigated this technique to refine the pose estimation of the optical outside-in tracking in the context of a bachelor's thesis [Wol15]. Since the position of the camera is estimated with high precision, only the orientation provided by the tracking system is expected to have higher variance. In our case, the particle-based approach was applied on the coarse model of the reconstructed scene which was sufficient to increase the overall tracking precision as shown in Figure 3.10.

3.4 Radiometric Calibration

Having extrinsic and intrinsic parameters allows the insertion of virtual objects with correct geometric registration. However, we also need reliable information about the radiance of the surface elements visible in the camera image to allow for capturing the illumination and estimation of material properties. Unfortunately, current imaging devices are not able to capture the full *dynamic range* of light present in the real world. The dynamic range of camera is defined as the ratio of the radiance that just leads to saturation of the sensor and the radiance that is just distinguishable from black level noise [DW00]. The range of conventional LDR images, which store colors as one byte per color channel per pixel, is around two orders of magnitude. In contrast, the sun at noon is about 100 million times brighter than starlight [Fer01] (see also Figure 3.11). When taking a picture, one task of the photographer is to select a suitable range that covers the interesting levels of brightness in the scene. This selection is described by another quantity referred to as *exposure* X measured in $[\mathrm{J/m^2}]$, which is defined as the product of irradiance E, that arrives at the sensor (or the film), and the exposure duration Δt:

$$X = E\,\Delta t. \tag{3.3}$$

In photography, the exposure is influenced by several parameters in addition to the exposure duration, which is also referred to as *shutter speed*. To illustrate these influences, the exposure can be considered as the number of photos that hit the sensor. This number is influenced by the lens and the aperture, where the latter can be manipulated to control the size of the opening, through which photons enter the system. The shutter speed directly controls Δt and thereby influences the duration in which photos can reach the sensor. To compensate for low light situations, the ISO sensitivity of the sensor is used to amplify the signal produced by photons hitting the surface. Aperture, shutter speed and ISO value, can be traded to achieve or avoid motion blur, depth of field and image noise. All three settings are designed to increase or decrease the exposure by factor 2, which makes the setup easier. A typically scenario is to fix one or two of the parameters and let the camera software configure the

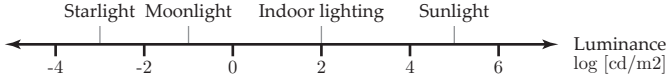


Figure 3.11: The Dynamic Range in the Real-world Environment
Typical real-world scenes cover a dynamic range of several orders of magnitude. An indoor photograph with the sky visible through a windows, is a common example that cannot be covered by the dynamic range of a single LDR image. Image courtesy of Ferwerda [Fer01].

(a) -3 stops (b) -1 stops (c) +1 stops (d) +3 stops

Figure 3.12: Exposure Series for Photometric Calibration
To recover the response curve of a camera, a series of images with different shutter speed is captured, while all other parameters are fixed. Here, we show images with exposure steps of 2 stops. Cameras usually allow steps of size $1/3$ stops.

free one to achieve a properly exposed image. In that case, there is another parameter, referred to as *exposure compensation*, to allow the user to take brighter or darker images. Typical values are in the range of -3 to $+3$ EV with steps of $1/3$ EV, where $+1$ EV again results in a doubling of the exposure.

When using a camera to measure the radiance of a real-world surface, we assume that all settings can be modeled by Δt. Hence, the exposure depends only on the relationship in Equation (3.3). Further, we can recover the irradiance from known exposure and exposure duration, while the irradiance is assumed to be proportional to the radiance in the scene [DM97]. This results in an estimation of the radiance up to a constant scale factor. In the case of AR, this scale can be ignored, but if the scenario requires absolute radiance measurements, a luminance meter can be used.

CAMERA RESPONSE CURVE An image acquired by a camera however does not store variations in exposure directly. Instead, a non-linear function is applied that maps exposure to pixel value Z:

$$Z = f(X) = f(E\,\Delta t)\,. \tag{3.4}$$

This function is referred to as the *camera response curve* f and mimics the response of film. It is usually not published in the specifications, as it is responsible for the characteristic look of images taken by a certain camera and thereby considered as a corporate secret. To relate pixel values Z to exposure X and eventually to irradiance E, we need to reconstruct the camera response curve f. Therefore, the method of DEBEVEC and MALIK is used [DM97], which relies on a series of images captured with varying exposure. To simplify the process, a static scene photographed from the same vantage point, so pixels of different images depict the same surface element. It is common to fix the aperture, as changing the aperture will influence depth of field and vignetting. To reduce the noise level, the ISO value is also fixed at a low setting. The shutter speed is adjusted automatically and the exposure compensation is changed to get an image series with varying exposure (see Figure 3.12).

Given the series of P pictures and N pixels that show the same surface in all images a set of relationships can be derived from Equation (3.4):

$$Z_{ij} = f(X_{ij}) = f(E_i\,\Delta t_j)\,,$$

where i is the i^{th} of N pixels and j is the j^{th} of P picture of the series. While assuming that f is monotonic, it can be inverted:

$$f^{-1}(Z_{ij}) = E_i\,\Delta t_j.$$

These equations allow to solve for the values f^{-1} and N values E_i. Since Z can take only a finite number of different values, e.g., 256, the size of the system is limited. The number of pixels N is also limited depending on the resolution of the images. The authors state that $N = 50$ is more than adequate for $P = 11$, while the selected pixels should be evenly distributed over the pixel values Z and homogeneous areas are preferred to avoid sampling issues. For details including the weighting of samples and the solving, I refer to the paper of Debevec and Malik of [DM97]. The result of the method is the inverse response curve represented as lookup table that allows to recover the relative irradiance for images and their corresponding exposure duration:

$$E = \frac{f^{-1}(Z)}{\Delta t}.$$

For the given camera setup, this curve is constant and independent of the scene. It allows to fuse several LDR input images into a HDR image. Therefore, the relative radiance of individual image areas that have not been under-saturated or over-saturated are weighted and averaged. This leads to a composition with an extended dynamic range and floating point values per pixel and color channel. A simple weighting function suggested by Debevec and Malik favors mid-range values:

$$w(z) = \begin{cases} z - Z_{min} & \text{for } z \leq 1/2\,(Z_{min} + Z_{max}) \\ Z_{max} - z & \text{for } z > 1/2\,(Z_{min} + Z_{max})\,, \end{cases}$$

where z is the LDR input value and Z_{min}, Z_{max} define the range of the input, e.g., $Z_{min} = 0$ and $Z_{max} = 255$.

Mitsunaga and Nayar presented a similar approach called Rascal that is able to handle imprecise estimations of the exposure steps used while capturing the image series [MN99]. In contrast to the method described above, the result of their method is a polynomial that approximates the response curve. The group around Shree K. Nayar collected a large database of recovered response curves and proposed an empirical linear model for camera response that is able to represent a wide range of response functions with only 4 coefficients [GN04].

However, for the approach in Chapter 5 and 6 we use the pfstools[14] that are based on a probabilistic formulation of the problem by Robertson *et al.* [RBS03], which, as they state, offers an improvement over the technique of Debevec and Malik. In our experiments both methods produced reasonable results.

An alternative work flow based on raw formats, with 11 to 13 bit per channel, can also be applied. In that case, the raw images are assumed to be in a linear color space, which simplifies the response curve. For mobile devices or when using webcams, this option is typically not available, at least not when using the live camera streams provided by the currently available APIs.

GAMMA CURVE In contrast to a photometric calibration, it is also common to assume that response can be modeled by a simple exponential curve:

$$Z = f(X) = X^{\frac{1}{\gamma}}$$
$$X = f^{-1}(Z) = Z^{\gamma}.$$

14 Max-Planck-Institut Informatik. *PFScalibration:Photometric Calibration of HDR and LDR Cameras*. Project Website, 2005.

Typically, a value of 2.2 is used for gamma, as this roughly corresponds to brightness perceived by humans, known from the *Weber–Fechner law*. It is also very similar to the curve that maps linear colors into sRGB color space, which is typically used to store LDR images.

3.5 VISUAL DISPLAYS

This section is again based on the book of SCHMALSTIEG and HÖLLERER and gives an overview of different visual displays and their key properties. For a detailed discussion on the individual topics as well as on non-visual displays, I refer to their book [SH16]. After describing the three fundamental methods to blend virtual objects into the real world and showing examples of the most prominent display types as well as their properties, the applicability of my own approaches to these types is discussed.

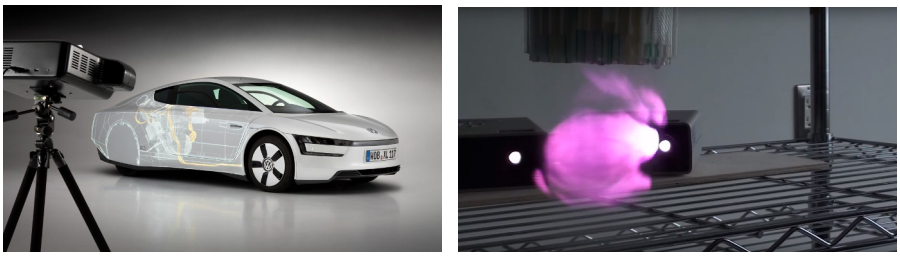
3.5.1 *Methods of Augmentation*

OPTICAL SEE-THROUGH These devices are usually designed as glasses or visors, like the MICROSOFT HOLOLENS (see Figure 3.13a). They consist of a partially transparent element that allows to directly observe the real-world environment by looking through it. This element also allows to display virtual content and thereby generates the desired blending. Displaying virtual elements can be realized for instance by projecting them from the top or sides onto the partially transparent element, that reflects the projection towards the eyes and thereby combines real and virtual content.

VIDEO SEE-THROUGH The second class of devices involves one or more cameras that capture the view into the real world. These images are used as background for composing augmented images, that are shown on one or more conventional opaque displays. Hence, the combination is handled by software rather than special optics and is therefore simpler to realize compared to Optical See-Through (OST). This class is currently the most dominant because hand-held devices, like smartphones, that are used for AR, follow that principle. But there are also head-mounted devices, like the VUZIX WRAP 1200DXAR that offer a binocular stereoscopic view into the augmented scene (see Figure 3.13b).

(a) Optical See-Through (b) Video See-Through

Figure 3.13: Optical and Video See-Through Devices
Examples of head-mounted devices: the MICROSOFT HOLOLENS (a) and the VUZIX WRAP 1200DXAR (b). Image courtesy of MICROSOFT and VUZIX, respectively.



(a) Projection on Real-world Objects (b) Fog Display

Figure 3.14: Spatial Projections
Examples of spatial AR solutions: a projector system that displays view dependent virtual elements directly on the surface of a real-world car (a) and a fog display prototype for projections in free space (b). Image courtesy of Volkswagen AG and Yagi *et al.* [Yag+11], respectively.

SPATIAL AUGMENTED REALITY Here, no optical combiners or electronic screens are required, as the augmentation is directly applied to the 3D environment. Spatial holograms, known from science fiction, that do not require glasses for instance are an idealistic example for a Spatial Augmented Reality (SAR) system. However, there are working prototypes existing today. *Spatial projections* for instance use light projectors that display virtual elements on real-world objects or fog displays that allow for augmentations in free space (see Figure 3.14a and 3.14b).

3.5.2 *Properties of AR Displays*

The design of display devices involves trade-offs between properties where each peculiarity has pros and cons. One of the features is the support of *stereoscopy*. Since humans are able to perceive binocular depth cues, a binocular device provides higher immersion than monocular systems, at the cost of a more complex and more expensive setup.

Another difficult problem is the visual *focus*. The human eye and real cameras have a limited depth range in which objects appear sharp. Outside of this range, visions gets more and more blurry. Computer generated content, that is rendered using a pinhole camera, is perfectly sharp at any depth. This results in a discrepancy that has to be addressed. Therefore, eye tracking can be used to anticipate the point of focus and the corresponding depth to alter the virtual elements.

As discussed earlier, *occlusion* is an important feature of geometric registration. For OST and Video See-Through (VST) systems, z-Buffers can be used to handle occlusion correctly. For SAR however, occlusion depends on the position of the observer, which is difficult to handle for spatial projections for multiple users in the presence of potential blockers. In the scenario shown in Figure 3.14a, another dependency on the vantage point can be seen. When the augmented elements are not located directly on the surface of the real-world receiver of the projection, their perceived position will vary when the user moves. Even for holograms a related problem exists. Specular effects on the surface of virtual objects depend on the position of the observer. In a multi-user scenario it will be difficult to present correct reflections for all participants.

The next properties discussed by Schmalstieg and Höllerer is *resolution* and *refresh rate*. Spatial resolution directly correlates with the visual quality of the produced images. Especially for head-mounted devices, the resolution is still critical, as the angular resolution of the human eye is not jet reached for displays located close to head. It is desired to achieve temporal resolution of 60 Hz in most scenarios. Objects in fast motion can require higher frame rates to be displayed without unintended blur. While most displays are capable to display images at rates of 60 Hz, there is usually a trade-off between performance and quality to cope with the computational limits during the image generation.

The *field of view* that is available for augmentation is another important aspect. OST devices, like the Hololens, currently have a rather limited field of view. This results in hard boundaries at which the virtual objects disappear when they leave the center of the view. For hand-held devices, that are used as magic lenses, the field of view depends on the size and the distance of the screen. For smartphones and tablets, the field of view for displaying virtual objects is small. Hence, the field of view of the camera becomes the limiting factor, especially, when tilting the device. At a certain degree of tilt, no background image is available anymore.

The next property is the *viewpoint offset*, which describes the spatially discrepancy between the camera positions at which the real and the virtual images are created. While OST devices can be calibrated, there are physical limitations for VST systems. Considering a hand-held device, which uses the back-facing camera to capture a real-world image that is augmented and presented on the display. To allow for a correct magic lens effect, the display can be interpreted as a virtual image plane, which needs to lie in the cameras view frustum. As this is not possible, there has to a spatial offset which leads to discrepancies, especially, when observing nearby objects. To address this issue, Schöps *et al.* recently presented a view correction based on a temporally consistent inpainting [Sch+17]. In the case of a head-mounted VST, like the one in Figure 3.13b, the camera would need to be located at the position of the observers eyes to capture an image from the same vantage point, showing the same perspective occlusions. An array of cameras that captures a 2D light field could be used to address that issue.

Another issue is *latency*. To avoid cybersickness while moving in AR and VR environments, it is important to reduce the delay between the movement of the user and the reaction of the visual representation to a minimum. One drawback of head-mounted VSTs is the delay, produced by capturing the background image and displaying it on the screen. Without any augmentation, this delay is notable and can feel uncomfortable. To keep virtual and real objects aligned, it is possible to further delay the input video stream when the rendering of virtual objects is not jet finished. This however, makes the problem worse.

Courtesy of Schöps *et al.* [Sch+17]

There are more relevant properties. *Brightness* for instance, which mainly concerns OSTs, as the display brightness needs to compete with the brightness of the real world in the background, and *contrast*, which is usually a problem related to the dynamic range of the involved image sensors. Cameras and other optical elements like lenses used for head-mounted devices can also show *distortions* and *aberrations*, especially when using low-cost components.

Courtesy of Itoh *et al.* [IHS17]

The two last aspects are *ergonomics* and *social acceptance*. Depending on the form factors of the device, it might not be suitable for every use case. Heavy head-mounted devices or hand-held devices that must be held at eye level cannot be used over long periods of time. This is closely related to social

acceptance. The acceptance depends on the form factor in terms of comfort but also in terms of a more or less pleasing appearance. In contrast to VR, augmented reality applications are "more social" in a certain sense, as it involves the surrounding. Users are not decoupled from their environment and other people. However, problems can arise because of the cameras that observe the environment. Even subtle designs, like the one of the Google Glasses[15], cannot prevent other people from feeling observed. Is is also likely to be perceived as rude when pointing a camera or smartphone at people to get augmentations with information about the person. Because of these issues, it could be possible that augmented reality will mainly be used in professional scenarios rather than day-to-day life.

3.5.3 *Visual Displays in this Thesis*

The techniques presented in this thesis mainly focus on any kind of video see-through devices. Typical form factors are *head-mounted displays*, *hand-held devices* and *stationary displays*. In the presented scenarios, we are using hand-held VSTs that act like *windows* into the mixed reality world [FZC93]. By adjusting the final image composition, optical see-through devices could also be targeted. However, depending on the technical realization of the device, it might not be possible to display elements that are darker than the background, because during the composition of the final image, light can only be added. To render shadows and dark virtual objects correctly, the display needs to be able to reduce the opacity for individual background pixels for instance. A prototype for such a display was presented by Itoh *et al.* The solution however, is not straightforward, as the occlusion mask appears blurred because the additional Liquid Crystal Display (LCD) panel, used to control the brightness of the incident real-world light, is out of focus [IHS17].

Extending the techniques to support stereo rendering should be straightforward by applying methods known from VR. Except the effort for rendering a pair of augmented images instead of a single one, the selection of the focus point and the simulation of depth of field are examples for additional tasks in such a binocular setup.

In contrast, SAR is different. Especially for multi-projector systems, image generation and spatially registration imply a different presentation pipeline. The core elements of the specific rendering techniques however, can be used nevertheless. Another issue in SAR is the representation of shadows since projectors are not able to subtract light from a real scene.

15 X. *Glass*. Project Website, 2017.

4

OVERVIEW OF COHERENT AUGMENTED REALITY

For many years, visual coherence in AR is one of the research topics in focus of the computer graphics community. With the farther goal of photorealistic augmentation, various methods have been developed to insert virtual objects into a view of the real world so that they blend in seamlessly. The foundation of this consistent rendering and seamless blending is a reliable reconstruction of the real-world environment, which includes the geometry of objects, the size, shape, position and the intensity distribution of light sources, as well as the reflectance behavior of the real-world surfaces. The first part of this chapter provides an overview of the related works, that address this reconstruction. Section 4.2 then covers *Differential Rendering*, the basic concept, that is used to create coherent augmentations based on light transport simulations. The third and last section gives and overview of the related works in the area of interactive coherent AR rendering. Descriptions of the individual works allow a classification of the own approaches in the context of the current state of the art and emphasize to possibilities of future research in the field, because many subproblems are not yet solved satisfactorily.

4.1 ENVIRONMENT RECONSTRUCTION

This Section provides an overview of the areas of environment reconstruction that form the basis for coherent AR rendering in this thesis. The described concepts and methods should be seen as related work of this thesis and are not a complete survey by any means.

4.1.1 *The Plenoptic Function*

The idealistic goal of the environment reconstruction is to gather knowledge about the incident radiance from every direction at every point in space and time. ADELSON and BERGEN describe the overwhelming amount of information by the so called *plenoptic function*, Equation (4.1), and provide a vivid explanation on how to interpret it:

$$P(\boldsymbol{\omega}, \lambda, t, \mathbf{x}) = P(\theta, \varphi, \lambda, t, \mathbf{x}_x, \mathbf{x}_y, \mathbf{x}_z)\,. \tag{4.1}$$

> *The world is filled with light, and the structure of this light is determined by the physical arrangement of the materials that fill the world.* [...] *For a given wavelength, a given time, and a given viewing position in space, there exists a pencil of light rays passing through the viewing point. Each ray has an intensity, and the collection of rays constitutes a panoramic image. This panoramic image will vary with time, viewing position, and wavelength.*
>
> — ADELSON and BERGEN [AB91]

In Equation (4.1), the wavelength is denoted as λ, time as t and the position of the viewing point as $\mathbf{x}$. The direction of a ray, passing the viewing point, is given in spherical coordinates, θ and φ. In our notation, the direction can also be described as an infinitesimal solid angle $\boldsymbol{\omega}$ pointing to that direction. Storing the information represented by this seven-dimensional function is simply not feasible, because as the authors state:

> *Such a complete representation would contain, implicitly, a description of every possible photograph that could be taken of a particular space-time chunk of the world* [...].
>
> — ADELSON and BERGEN [AB91]

Consequently, only fractions of this data are captured and stored using a representation that fits the needs of the application as well as the capabilities of the acquisition method used. As described earlier, it is common to ignore the wavelength and use three independent discrete RGB channels instead, at least in most areas of computer graphics. The time dimension can also be neglected if the environment is assumed to be static. Even though many techniques are designed to deal with dynamic environments, they usually do not require the availability of the information for every point in time. In general, the information of the current time step (i.e., live data) is sufficient. Since the area of interest is limited to a certain space for most applications, it is convenient to reduce the covered space to the local scene. However, the amount information of this 5D subspace is still difficult to handle. In the following, this subspace will be referred to as a *dense light field*.

Limiting the spatial extent to two dimensions is an elegant way to further reduce the domain to four dimensions. This leads to a planar representation that is often just called *light field*. Even though *light field* is the most often used term, we refer to it as *light slabs* to avoid confusion with the more general term, describing the 5D *(dense) light fields*. The basic idea of light slabs is to store the view-dependent radiance for each point on a plane. Assuming that there are no occluders and that absorption, scattering, and diffusion are negligible in the air, the radiance information along an arbitrary direction is constant [Ash93, LH96]. Considering a ray traveling in that direction and intersecting a plane located in such a *free space*, the radiance acquired by those rays can be stored in a two-dimensional field, indexed by θ and φ of the ray directions, at each intersection point with the plane. This results in a 4D structure.

Reducing the spacial domain to single point, leads to the well known *light probes*. This 2D representation stores radiance with respect to the viewing direction. Technically, they can be described as photometric space-time samples of the plenoptic function, highlighting the metaphorical usage of the term "probe". Details on the three representations will be discussed in Section 4.1.2.

All these representations are instances of the group of *light-based models*, as the information contained is purely light-related. Approaches that project light information on a reconstructed geometry or a coarse approximation are also classified as light-based. Debevec for example, projected a light probe onto a simple cuboid serving as coarse geometry approximation [Deb98]. This allows to approximate the position dependent plenoptic function by intersecting rays with the geometry despite the fact that only one light probe was used to capture the light information.

Incorporating knowledge about the geometry and materials present in the scene leads to *material-based models*. To capture them, a more complex reconstruction procedure is required. After reconstructing the geometry of real objects, *inverse rendering* methods are used to estimate material properties and light information from real images or image sequences. All topics are discussed more detailed in the following sections.

The result however, is the foundation of a coherent AR rendering. Material properties of real objects are required for computing simple shadows, cast by virtual objects, and allow the correct simulation of complex light interactions. The quality of the estimated properties directly influences the possible features, visible in the resulting augmentation. This means that methods for creating more complex and more accurate environment models are required, which results in an increasing amount of effort. The users however, should not spend more time on the issue as this would result in a decrease of acceptance and applicability. Similar to the representation of the light, there are multiple options to model and store the acquired information. Coarse approximations can be acquired faster but may not allow for some lighting effects.

The third class of models are the *image-based models*. These models describe the scene implicitly by a collection of images or photographs (see Page 75).

Given a material-based model, instances of the other classes can be created by computing illumination solutions for it. To get an image-based model a finite set of images has to be rendered. For a light-based model, like a dense light field, the computation can be extremely expensive because a converged solution for every point and viewing direction of the domain has to be reached. Deriving a material-based model from one of the others, again requires *inverse rendering* methods.

4.1.2 *Light Representation*

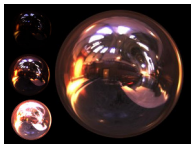
HIGH DYNAMIC RANGE IMAGING (HDRI) To be able to acquire real-world radiance and to eventually perform physically-based rendering, it is important to capture radiance with the full dynamic range of the real scene. DEBEVEC and MALIK describe how to capture such a high dynamic range image from multiple low dynamic range images with different exposure [DM97] (see Section 3.4).

In case the capturing of an exposure series is not possible, e.g., while working with legacy footage, there are alternatives to acquire HDR images from LDR input. One is applying Inverse Tone Mapping Operators (iTMOs) that expand the dynamic range of the input images. Tone Mapping Operators (TMOs), like the one of REINHARD *et al.* [Rei+02], are used to map HDR colors to a low dynamic range of 8 bit, while maintaining details in bright as well as dark regions. Thus, the mapped images can be properly shown on a LDR display. The idea is to apply the inverse of a TMO to reason about the saturated areas in given LDR images. Therefore, the input is linearized using the camera response curve and overexposed areas are extrapolated. Because this involves many assumptions and heuristics, the resulting HDR images are only approximated and for IBL as well as for coherent rendering, more direct measurement techniques are preferred. However, it is a valid and promising approach for scenarios with certain restrictions on the hardware setup, too. For an overview on this topic, I refer to the survey of BANTERLE *et al.* [Ban+09], who also authored one of the investigated iTMOs that can be used for IBL and thereby for rendering in AR.

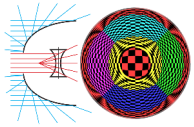
Of course, manually specifying the intensity of overexposed regions is an option, too. Besides the additional user effort, it is far from easy to select the correct intensities.

LIGHT PROBES Incident light is represented as an omnidirectional image of the real scene measured at a certain position. This image is also referred to as *light probe* and is stored as an *environment map* in one of various parametrizations, e.g., lat-long panoramic image, cube map or two hemispherical projections. Each pixel of this environment map stores the average incident radiance over the solid angle subtended by that pixel. Light probes are typically used as input for IBL (see Section 2.7.4). They are captured using a mirror sphere as presented by DEBEVEC [Deb98] or omnidirectional cameras, which are usually covering only one hemisphere or less.

Even for an acquisition with a mirrored spheres, various details have to be considered [Rei+10]. The photographer and camera as well as the support of the sphere are visible in the images, so the camera has to be positioned relatively far from the sphere to capture, but the sphere itself needs to cover a large portion of the frame and needs to be in focus. The region behind the sphere, which is reflected on a small area close to the silhouette, is not well represented in the images. To overcome this issue - and also the visible photographer – a second image sequence from a view point rotated by 90° can be used to fill the corresponding regions. The sphere itself is also not perfect. It is hard to manufacture ideal spheres. Usually they have tiny scratches causing non-specular reflections and their surface is not reflecting the entire incident light. However, the latter can be calibrated.



Courtesy of DEBEVEC [Deb98]



Courtesy of UNGER and GUSTAVSON [UG07]

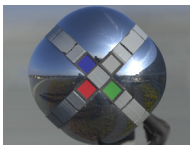
UNGER and GUSTAVSON suggested a clever alternative to mirrored spheres that allows to capture an entire light probe with single image sequence and

better angular resolution for the areas behind the sphere. Therefore, they created a passive device consisting of a mirror that swept around a lens in the center [UG07].

A variety of other probe-based methods have been presented. Using a diffuse sphere allows to estimate the brightness of single directional light source. In combination with a second image of a mirrored sphere, it allows to estimate a single light source, which was overexposed in the mirrored sphere image, by fusing the information from the diffuse sphere [Rei+10]. This method is especially useful in outdoor scenarios with bright sun light.

Courtesy of Reinhard *et al.* [Rei+10]

Another example is the single shot light probe of Debevec *et al.* [Deb+12]. After taking an image with a single exposure, saturated light source intensities can be estimated by solving a linear system using samples of diffuse stripes.



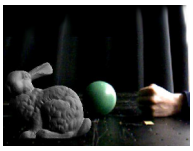
Courtesy of Debevec *et al.* [Deb+12]

A known planar image can also be used as light probe. Pilet *et al.* estimated the distant directional light by observing a known diffuse image from multiple directions over time [Pil+06]. While the planar image can also be used for tracking and calibrating, the different orientations allow to reason about the incident lighting, as the observed brightness relates to the total irradiance incident from the hemisphere pointing towards the plane normal.

Light probes, that contain only light of low frequency and no strong peaks in brightness, can be compressed into SH basis and thereby used for fast rendering based on PRT, as discussed in Section 2.7.6. Compressing incident light using SH works for various objects, i.e., not only for spherical probes. When the shape and the (diffuse) reflectance of the object is known, SH coefficients, describing the low-frequency environment illumination, can be derived and rendered. This was done for a users hand [YKK12] by Yao *et al.* and for the face by Knorr *et al.* [KK14], for instance. While Yao *et al.* used a Kinect sensor, the approach of Knorr *et al.* required no special equipment, as they learned an average model of the reflectance on a human face.

Courtesy of Pilet *et al.* [Pil+06]

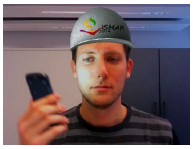
Another unique solution was presented by Calian *et al.* [Cal+13]. Their *shading probes* can be used to capture the environment illumination in an orthogonal basis directly. A sphere-like, 3D-printed object partitions the sphere – and thereby the incident radiance – into N disjoint sections. The acquired radiance per patch can be used in the standard PRT framework directly.



Courtesy of Yao *et al.* [YKK12]

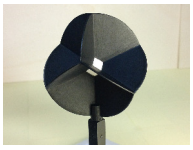
Acquiring the light probe using an omnidirectional camera is more straight forward if the camera and the lens system are calibrated well as described in Section 3.2. The resulting hemispherical images can be sufficient, e.g., in cases where the camera is placed close to a known flat surface, like on a table. However, it is of course also possible to create a light probe by stitching several (perspective) images [Che95].

Recently, several consumer devices[16] have been released that are able to capture 360° video streams. This can help to improve usability and to overcome the limited field of view of conventional wide-angle lenses.



Courtesy of Knorr *et al.* [KK14]

An overview of further probe-based light acquisition methods can be found in the recent survey of Kronander *et al.* [Kro+15]. For more details on how to achieve high quality light probes I refer to the HDRI book of Reinhard *et al.* [Rei+10] and the course notes [Rei+06]. For explanations from a photographers point of view the HDRI Handbook of Bloch is a good reference [Blo12].



Courtesy of Calian *et al.* [Cal+13]

16 E.g., 360fly, Ricoh THETA Series, LG 360 Cam or the Samsung Gear 360.

TIME-VARYING LIGHT PROBES Capturing real-time high dynamic range light probes over time is the next logical step towards dealing with interactive scene illumination. Since different methods exist, which require only one single shot to capture a HDR light probe, it is straightforward to capture a video stream instead of the single image.

WAESE and DEBEVEC applied natural density filters to the faces of a faceted lens in order to capture multiple exposures in a single frame [WD01].

TOCCI *et al.* presented a special video camera system that uses beamsplitters behind the camera lens and thereby divide the incident light to be processed by multiple sensors – in their case three. The resulting low, mid and high-exposure images are then merged into a HDR video stream [Toc+11]. Later, KRONANDER *et al.* presented a unified and flexible framework for fusing multi-sensor HDR video data [Kro+14].

Without this kind of special equipment, capturing and fusing images with varying exposure is the main challenge. In a dynamic environment, the single images have to be acquired in a short period of time or the temporally and spatially discrepancy becomes an increasingly difficult challenge.

Courtesy of WAESE and DEBEVEC [WD01]

SPATIALLY-VARYING REPRESENTATIONS An intermediate step between the distant light assumption and spatially-varying light fields can involve the analysis of multiple light probes.

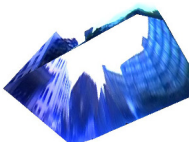
With the help of two HDR light probes – 180° fish eye images, SATO *et al.* were able to semi-automatically reconstruct a coarse triangular mesh of the distant scene from feature points visible in both images [SSI99]. By projecting the acquired camera images onto the mesh, the radiance is stored in textures for AR rendering.

Courtesy of TOCCI *et al.* [Toc+11]

For their approach based on stereo light probes, CORSINI *et al.* do not reconstruct such a mesh. However, they also find correspondences in the spherical images to extract the position of light sources in world space. Experiments show, that the technique is able to resemble area light sources by a set of point lights to a high degree of accuracy [CCC08].

Courtesy of KRONANDER *et al.* [Toc+11]

LIGHT FIELDS As described earlier, light fields are used to describe a subspace of the plenoptic function. At this point, we assume that this subspace is a 5D domain, describing radiance depending on the position in space $\mathbf{x}$ and the viewing direction $\boldsymbol{\omega}$. Hence, light fields are used to store spatially varying illumination, which drastically improves renderings by allowing for shadows and reflections (see Figure 4.1). In the literature, there are two kinds of data structures that are called light field. Both are closely related but describe slightly different information as described by Kronander et al. [Kro+15].



Courtesy of SATO *et al.* [SSI99]

Lights fields in the context of photography, store outgoing radiance of the photographed scene, i.e., reflected or emitted radiance. One prominent example is the *Lumigraph* by GORTLER *et al.* [Gor+96]. By capturing light slabs arranged as a box around the object of interest, it allows to render the object from arbitrary viewpoints outside the box showing correct view-dependent illumination without explicitly reconstructing the geometry or materials. Recently presented light field cameras[17] allow consumers to make use of the light slab rendering presented by LEVOY and HANRAHAN [LH96]. Compared to a conventional image, the light field captured from a small plane acquires enough spatial information to allow for slight movements of the camera or refocusing

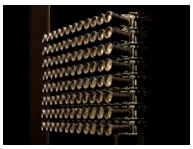
Courtesy of CORSINI *et al.* [CCC08]

17 E.g., the Lytro light field cameras.

Figure 4.1: Spatially Varying Illumination
Comparison of simulations using a single light probe and a spatially varying illumination description. Image courtesy of Unger *et al.* [Ung+13].

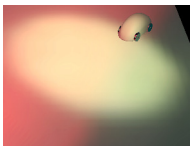
objects after the image was already captured. The capabilities of a large scale system, using this technique, have been presented earlier by Wilburn *et al.* [Wil+05]. They showed a grid of 100 cameras in different arrangements allowing various applications from light field capturing to high speed acquisition or super resolution HDR. By fusing information from different viewpoints they are also able to remove occluders.

In the context of AR or when rendering virtual objects using real-world illumination, an Incident Light Field (ILF) is used to store the inbound radiance in the area, where the virtual object should be placed. The captured illumination contains radiance in the full dynamic range to allow for correct physically-based rendering, which is why this representation is the one we are mostly interested in. To emphasize the connection to rendering and Kajiya's rendering equation (2.12), incident light fields are denoted as $L(\mathbf{x}, \boldsymbol{\omega}_i)$ instead of using the notation of the plenoptic function. In the following, the usage of the term *light field* implicitly means (dense) ILF.



Courtesy of Wilburn *et al.* [Wil+05]

Courtesy of Unger *et al.* [Ung+03]



Courtesy of Löw *et al.* [Löw+09]

As stated earlier, the volume captured in a light field is limited to a certain spatial extent. Naturally, this extent is defined by the scene, but to cope with the amount of data and the complexity of the capturing process, the measurements are often done in a significant smaller domain. An overview of several approaches to capture spatially varying illumination along 1D paths, in 2D planes and in 3D volumes of interest can be found in the report of Kronander *et al.* [Kro+15]. Since most of the approaches were presented by the group of Jonas Unger, I also refer to his thesis for details about capturing, processing and rendering light fields [Ung08]. The basic concept of these approaches consists of a light probe measurement phase and a reprojection phase. In the first phase several light probes in the area of interest are captured. Their number varies from a few up to thousands. The images are then re-sampled and stored as ILF-planes, i.e., parallel light slabs spanning the volume. Besides sequentially moving a pair of camera and mirror sphere to several locations, multiple spheres can be captured with one shot. Unger *et al.* suggested to use an array of 12×12 mirror spheres. In the same work, they investigated an omnidirectional camera mounted onto a translation stage, that allows to sample the incident light field within a plane [Ung+03].

Löw *et al.* proposed a structure that is able to directly store a compressed light field in discrete 3D grid [Löw+09]. Each cell of this grid stores the angular variation in the illumination compressed in SH. This resulting structure is very similar to *Irradiance Volumes* [Gre+98] and their extension by Christo-

PHER OAT, who also stored the irradiance data compressed in SH basis in a grid structure [Oat05, Oat06]. The regular structure of those grids and the orthonormality of the SH basis functions allows for fast evaluation on the GPU (see Section 2.7.6).

IMAGE-BASED MODELS Light fields as well as light probes are typically referred to as *light-based models* of the environment. However, the term *light field rendering* is often used in context of a more specialized field, the rendering of scenes represented by an *image-based model*. Sometimes, the term *image-based rendering* is also reduced to that narrow field. In contrast to a light field, that stores the radiance from or to all directions at all positions in space, an image-based model stores radiance from only a few directions at multiple positions in space. In practice, it consists of a set of photos, captured in the scene. These photos are usually stored in a graph along with the poses of the camera and they may contain pixel values that are transformed and truncated by the response curve of the camera [Deb98]. For rendering, a ray-cast into the scene is replaced by interpolating between images that fit origin and direction of the ray best. Even though the goal is to capture all containing objects in the scene from multiple directions, the acquired information is theoretically of lower dimensionality. Assuming a static scene captured in RGB, the light field is a 5D function of $\mathbf{x}$ and $\boldsymbol{\omega}$, while the image-based model is a (large) set of two-dimensional images, usually resulting from a sparse spatial sampling. For application in AR, the selection of cameras poses is even more difficult as not only the observer position but also the position of virtual objects has to be anticipated in order to acquire a sufficiently dense model.

MEILLAND *et al.* presented an approach to create such a set of HDR images for key-frames of a tracked RGB-D camera [MBC13]. Key-frames have previously been used for 6D tracking in large scenes, e.g., by KLEIN and MURRAY [KM07]. New key-frames are generated only when new parts of the scene become visible, which reduces the amount of dense data to store drastically. Otherwise, the model encoded in existing frames is improved. MEILLAND *et al.* applied that principle to light field capturing, by continuously projecting new LDR input images into the key-frames. This projection is possible because of a dense geometric model of the scene that is created using the depth sensor. By allowing the exposure time to change, the LDR color information can be fused to HDR. Because the observed radiance of multiple views is averaged in the HDR texels of the key-frames, the environment material is assumed to be Lambertian. Hence, the captured light field only stores view-independent data. For rendering virtual objects, environment maps are rendered at arbitrary positions to provide a light probe for image-based rendering.

Courtesy of MEILLAND *et al.* [MBC13]

LIGHT RECONSTRUCTION In contrast to the described techniques for capturing parts of the light field in the scene, it is also common to reason about the light conditions based on the shading of objects in the real scene. The process of extracting light parameters from shading is part of *inverse rendering* and covered later in Section 4.1.5. This however involves knowledge about the geometry of the scene, which is the topic of the next section. Before that, a few alternative approaches to acquire the light condition of a scene are mentioned.

Courtesy of MADSEN and NIELSEN [MN08]

GUESSING LIGHT CONDITIONS When the goal is to achieve a plausible rendering rather than having a correct rendering based on a decent light re-

(a) GPS-based Sun Position (b) Comparison to a Printed Bunny

Figure 4.2: Results of Sensor-based AR
Results presented in the bachelor's thesis, that investigated AR render based on sensor data and information from an online weather service. Image courtesy of Andrei Stein [Ste14].

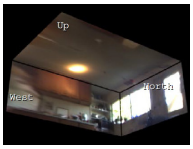
construction, it is possible to estimate light parameters that roughly match the current scene. Madsen and Nielsen showed that a simple camera image, containing real-world shadows along with the corresponding GPS location and the orientation provided by a digital compass, as well as the current time and date are enough to estimate parameters of a sky model in outdoor AR scenarios [MN08]. In the context of a bachelor's thesis of Andrei Stein, we investigated a related idea [Ste14]. Here, the sky-model parameters have been estimated by an image of a white sheet of paper, the current GPS location, compass, date, time and an additional cloudiness factor derived from an online weather service. Results are shown in Figure 4.2.

Lalonde *et al.* also used a weather information [LEN09]. They build a database of environment maps from over a million images of 1350 public webcams all over the world. These maps cover a large variety of different scenes at different times and in different weather conditions. By matching the illumination of a given input image with the database, a proper environment can be queried for shading virtual objects.

4.1.3 *Geometry Reconstruction*

A suitable geometric representation of the real environment is the foundation of any light simulation and coherent AR rendering. Depending on the technique used to compute this illumination solution, different levels of detail are required. When the position of virtual objects is known beforehand, it is sufficient to create a detailed model of the area of interest, the so called *local scene*, while the rest of the environment, the *distant scene*, can be represented by simple geometry like a box [Deb98]. Debevec introduced this partitioning based on the influence range of virtual objects rather than spatial relationships alone (more in Section 4.2), but at this point the basic idea serves the purpose. In a scenario where the user can move freely and place virtual objects at arbitrary positions, more complete knowledge about the scene is required for correct occlusion, shadowing and illumination at any location.

Courtesy of Lalonde *et al.* [LEN09]



Courtesy of Debevec [Deb98]

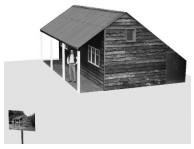
The goal of geometry reconstruction is having reliable information about surface positions and normals of all relevant real-world objects. Since triangle meshes are easy and efficient to handle by GPUs, they are the most common representation. Other forms, like volumes or point clouds, can also be used. Independent of the representation, a model as simple as possible enables us to achieve performance goals but an adequate detail is required to represent the real world objects in order to compute plausible shadows and occlusion.

In general, it is required to use a calibrated camera when deriving geometric information from images. While most approaches assume knowledge about intrinsics, the estimation of extrinsic parameters and the geometry reconstruction go hand in hand very often. The tracking of natural features for instance (see Section 3.3) is closely related to approaches for multiple images described in this section.

MANUALLY MODELING A valid approach to acquire a mesh representing the real environment is manually measuring the size of real world objects and their position in the scene to construct a 3D model using a Digital Content Creation (DCC)-tool[18]. This seems to be an exhausting task, but due to the fact that only a coarse approximation of the geometry is sufficient for simple AR applications, a manually created model has multiple benefits: The result contains an arbitrary level of detail. The meshes are manifolds and free of holes. A user is able to create separate meshes for individual objects which implicitly allows for rigid transformations in dynamic scenarios. The scenes used in Chapter 5 and 6, have been reconstructed manually. In the context of this work, another important benefit is the (manual) generation of uv-maps, which provide a unique mapping between 3D surface positions and 2D texture coordinates for efficiently storing scene information in textures. Figure 4.3 shows the resulting reconstructed geometry of our lab.

Courtesy of FOURNIER *et al.* [FGR93]

FROM SINGLE IMAGE Simple geometrical objects can be retrieved manually from a single image. FOURNIER *et al.* for instance, used cuboids only to represent the objects in the scene [FGR93]. These boxes can be created by selecting at least four points in a particular order. In general, such *object-based reconstruction* methods exploit parallel lines and faces, orthogonality and symmetry to define simple polygonal shapes, as mentioned by GROSCH [Gro07]. Because of the single view, constraints are required to resolve ambiguities. Therefore, the proposed systems are interactive and rely on user input, e.g., the ones proposed in [Heu98, SM99, Gui+00].



Courtesy of CRIMINISI *et al.* [CRY00]

Additional knowledge about repeating patterns in architecture, or objects of equal size like the height of windows, can also be incorporated into the reconstruction process [LCZ99, CRY00].

In some AR applications it can be sufficient to estimate a per pixel depth or even to classify a few layers at certain depths to handle occlusion between real and virtual objects. For these cases several techniques have been proposed to create pop-ups or billboards from single images that are placed in 3D space [Oh+01, Kan+01, Zha+02, HEH05]. For a visually coherent rendering however, a more complete description of the scene is important.

Courtesy of OH *et al.* [Oh+01]

In recent years more and more learning-based methods have been presented. One of them by HEDAU *et al.*, who recover the spatial layout of the scene from a single image [HHF09]. Combined with the billboard technique of KANG *et al.*, this layout enabled KARSCH *et al.* to augment legacy images by virtual objects after a certain amount of user annotations [Kar+11]. Despite their simple geometric model, the system can achieve plausible results.

Courtesy of HEDAU *et al.* [HHF09]

FROM MULTIPLE IMAGES More information is available when using more than one image as input. To exploit redundant information, we first retrieve feature correspondences in two or more images. This can be done manually by selecting pairs or tuples of the feature point visible in two or more images,

Courtesy of KARSCH *et al.* [Kar+11]

18 E.g., 3Ds MAX, MAYA, CINEMA4D or MODO.

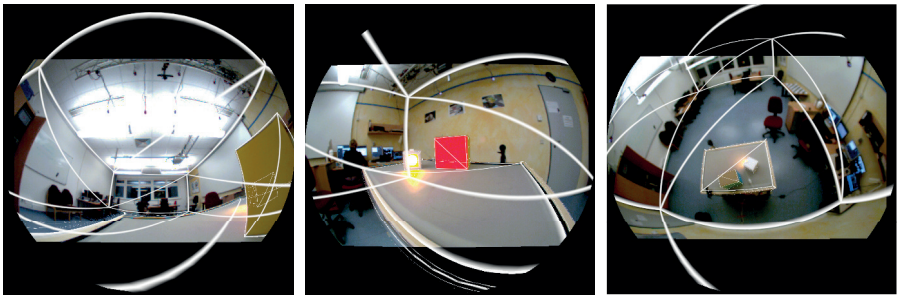


Figure 4.3: Reconstructed Geometry of the Lab
Real scene captured by three cameras simultaneously. The manually reconstructed geometry of the scene is overlaid as wire frame.

respectively. In the manual case, features are usually corners of objects or details in their surface texture. Highlights or other view-dependent properties cannot be used. When intrinsic and extrinsic camera parameters are known, a world space ray can be created for each selected 2D feature in each image. For the simple case of two images, the 3D positions of the corresponding feature can be triangulated by finding the points on the rays, where both rays come closest. In general, both rays are not intersecting exactly, so the triangulated position is defined by the midpoint of the shortest connecting line between the two rays. For more images, the simple midpoint computation needs to be replaced by solving an eigenvalue problem. Connecting the extracted 3D points to individual meshes can be done in an interactive iterative process. Enabling the user to specify individual constraints allows to easily create simple polygonal models [POF98]. Even those computer vision aided methods still require a immense manual afford of several hours [DRB97].

Establishing correspondences can also be done automatically. Therefore, vision-based methods are used to extract features in all images. The challenging part is the subsequent feature matching. Features detected in one image need to be assigned to features in a second image. For known camera parameters, this search is limited to points on or close to the epipolar line, but usually multiple candidates need to be tested. Integrated into interactive modeling systems, like the sketch-based tool presented by Sinha *et al.*, detailed models can be created in a few minutes [Sin+08].

If the relative transformation between the cameras is unknown, it can be retrieved from given corresponding features using the *Eight-Point Algorithm* [Lon81, Har97] or the *Five-Point Algorithm* [Nis04, LH06] and thereby the same principals of epipolar geometry. They allow to compute the *fundamental matrix*, which defines, for a given feature in one image, the epipolar line in the other image. This process is the foundation of calibrating stereo camera systems and forms the basis of the automatic matching described above.

Courtesy of Sinha *et al.* [Sin+08]

Drettakis *et al.* used an image mosaic to increase the field of view [DRB97], as panoramic images provide even more information. Several of these mosaics are then used for multi-view reconstruction.

Courtesy of Drettakis *et al.* [DRB97]

As these image-based reconstruction methods are not used for the own contribution in this thesis, I will not provide more detail or further examples and refer to books on multiple view geometry, e.g., the one of Oliver Faugeras [Fau93] or Hartley and Zisserman [HZ04].

USING DEPTH SENSORS Systems using time-of-flight laser rangefinders combined with conventional cameras to capture color information, like the one of NYLAND [Nyl98, McA+99] have been presented in the 1990s already. Before, CURLESS and LEVOY scanned the famous objects available in the *Stanford 3D Scanning Repository* [CL96]. At this time, the resulting 3D meshes have not been very suitable for the use in AR [Gro07]. One of the reasons was the amount of details captured in the point cloud data. Meshes, created directly from those point sets, often contain holes or are non-manifold. Even without these problems, the resulting meshes are of high tessellation and many of the approaches have been using light transport simulation based on radiosity, which does not perform well for very large numbers of patches. However, the most important point was the limited availability of such sensors.

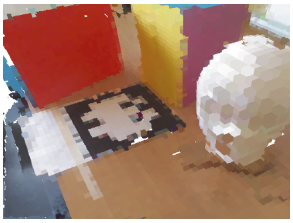
Recent advances in hardware made time-of-flight techniques attractive again [SH16]. Inexpensive consumer devices[19] provide RGB-D sensors to measure depth samples directly without the requirement of an triangulation. An important advantage, that allowed to overcome the earlier challenges, is the availability of powerful GPUs.

NEWCOMBE and IZADI presented *Kinect Fusion*, a technique that combines the acquisition of 3D volume containing the geometry with a camera pose estimation based on dense SLAM [New+11, Iza+11]. Registering the current depth image of the camera with the recorded geometry of the last frame allows to estimate and iteratively refine the new camera pose. By porting the Iterative Closest Point (ICP) algorithm [AHB87, BM92] to the GPU and assuming that it can be calculated in a few milliseconds, the algorithm can be approximated by image-space operations. This approximation is valid, because the spatial transformation during that short time is considered small. The depth samples are not stored directly to the volume, instead a *Truncated Signed Distance Function* is used to implicitly describe the containing surface. While integrating a new dataset, the distances are blended to get a smoother surface description with less noise than in the original depth image. Casting rays into the volume allows to extract depth images, e.g, as input for the ICP in the next frame or for other parts of the rendering pipeline. The downside of this technique is the memory consumption, especially when other properties like the surface color are stored, too. Since the volumes are dense, it is a trade-off between memory consumption, spatial extent of the volume in the scene and grid resolution. Since large parts of the volume are usually empty, the shortcoming of this dense volume is obvious and several extension have been presented to address the issue.

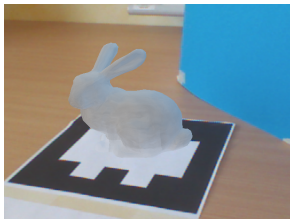
CHEN *et al.* for example increased the size of the acquired volume by storing the signed distance function in a hierarchy of three to four levels, where inner nodes without geometry are handled memory efficient [CBI13]. They also allowed to stream data between the CPU and GPU to overcome the memory limitations of GPUs. However, the size and position of the volume still has to be defined beforehand. Replacing the fixed sized volume by multiple smaller volumes, that are addressed by their (hashed) position, allows to acquire larger or more detailed scenes and to store dense volumes only in areas containing actual surfaces [Nie+13]. Because of the unstructured nature of the hash volume, there are no spatial constraints as long as memory is available for new block allocations and hash collisions are avoided or treated efficiently.

In the context of the master's thesis of CHRISTIAN SONDERFELD, we also experimented with voxel hashing [Son15]. RGB-D samples, generated by a

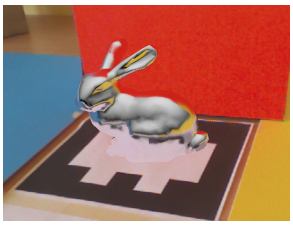
19 E.g., the ASUS XTION, MICROSOFT KINECT, INTEL REALSENSE or GOOGLE TANGO.



(a) Voxel Structure



(b) Diffuse Virtual Bunny



(c) Glossy Virtual Bunny

Figure 4.4: AR Rendering using a Spatial Hash Grid
Results presented in a master's thesis, that investigated spatial voxel hash grids to capture a dynamic scene for AR rendering. Image courtesy of Christian Sonderfeld [Son15].

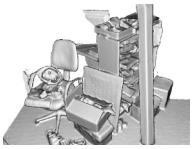
tracked Kinect sensor, are added directly to a 3D spatial hash grid. The system is able to handle moving real-world objects by voting down and eventually removing outdated voxels. Differential rendering (see Section 4.2) as well as the removal of occupied voxels that stay in conflict with the depth information in the current input image, rely on ray-marching. For rays not hitting any voxel, because of an incomplete reconstruction, a fallback environment is used, provided by an additional omnidirectional camera or by composing the color frames of the Kinect. A visualization of an acquired test scene as well results of the rendering are shown in Figure 4.4.

For the approach presented in Chapter 7, we use a spatial hash grid to control the density of samples across the regions, acquired using a mobile depth sensor. The position in a virtual grid with a cell size of 0.005 m is computed for each acquired sample. This position is hashed and only one sample is stored per hash, possibly discarding previous samples with the same hash. The replacement happens randomly based on a chance that depends on the number of collisions for that hash value.

Voxel hashing and further optimization allows to use the approach even on mobile hardware as shown by Kähler *et al.* in the InfiniTAM framework [Käh+15]. Dai *et al.* presented a distributed approach in which a mobile device is used for capturing RGB-D frames and a desktop PC for pose estimation and reconstruction. After streaming new frames to the PC, a globally optimized (bundle adjusted) pose is estimated in real-time. The framework also creates a 3D reconstruction of the scene that supports loop closure. Hence, the reconstruction is updated in real-time to create a high quality globally consistent geometric representation of the scene [Dai+17].

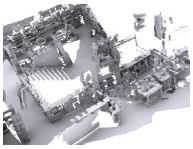
A different option to reconstruct larger scenes is to move the volume along with the camera in the sense of a ring buffer. While moving the device into a certain direction, the volume follows and the data stored in voxels on the other side is dropped. Before actually losing the data, a triangle mesh is created to represent the surface in that area [Whe+12]. Creating a triangle mesh from a 3D scalar field, like the signed distance function, is a well-known problem that can be solved using the *marching cubes* algorithm [LC87] to extract the surface that is implicitly defined at the zero-crossing. It could also be applied easily as post-process to all the presented volume based methods.

Another focus of recent research is to achieve compelling results without depth sensors [NLD11]. Here, the depth is derived from a single camera by using stereo information after slight movements. Using mobile cameras and



Courtesy of Newcombe *et al.* [New+11]

Courtesy of Izadi *et al.* [Iza+11]



Courtesy of Chen *et al.* [CBI13]

Courtesy of Niessner *et al.* [Nie+13]

Courtesy of Kähler *et al.* [Käh+15]

Courtesy of Dai *et al.* [Dai+17]

Courtesy of Whelan *et al.* [Whe+12]

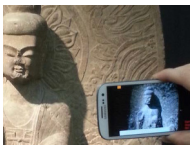
fusing data from other sensors also allows to estimate absolute scale and acquire (small) models using hand-held devices [Tan+13].

POINT-BASED The last group of approaches covered in this section capture the geometry in an unstructured point cloud. Even though the following methods rely on RGB-D data, most of them could also be extended to support moving monocular or stereo RGB cameras.

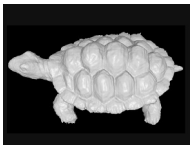
Transforming the acquired depth samples in a global coordinate system and storing them in a simple list is the most straightforward solution. Without a spatial data structure, there are no constraints on the extent of the scene and assuming the required transformation is known, basic operations like adding, removing and rendering points, map well to the GPU architecture. Unfortunately, the transformation is not known in most scenarios. Hence, it is required to use an additional tracking method or to combine the camera pose estimation with the reconstruction. The scanning system presented by Rusinkiewicz *et al.* for example, uses ICP to align the point data of different frames [RHL02].

Courtesy of Newcombe *et al.* [NLD11]

An issue closely related to tracking is the problem of accumulated drift over time. Since the camera pose is estimated based on the model, which is currently reconstructed, small errors sum up over time. When the recording device is reaching a previously captured area, the most recent captured geometry does not match the old one, which results in gaps. To overcome this issue – and close the loop – a *bundle adjustment* or, for larger scenes, a *pose graph optimization* is performed. Weise *et al.* extended the idea of Rusinkiewicz *et al.* and incorporated an online loop closure to address the accumulated registration errors in their in-hand scanning system [Wei+09]. For larger scenes, Henry *et al.* combined visual feature tracking with depth sensing [Hen+12]. Their extended ICP algorithm matches features based on RGB-D data. By applying a *sparse bundle adjustment* for global loop closure, they are able to acquire complex scenes, e.g., circular hallways with several rooms.



Courtesy of Tanskanen *et al.* [Tan+13]



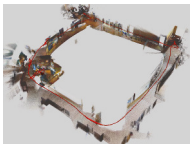
Courtesy of Rusinkiewicz *et al.* [RHL02]

Related to the *Kinect Fusion* approaches, covered in the previous section on point clouds, Keller *et al.* presented a fusion approach based on a flat point-based representation. Without the need of converting into the signed distance volume, they directly work on the point cloud to generate a denoised depth image, estimate the camera pose and update the point cloud [Kel+13].

Courtesy of Weise *et al.* [Wei+09]

Whelan *et al.* showed that the methods for loop closure used in SLAM can also be applied to fusion approaches, which leads to non-rigid deformations of the point cloud that adapt detected loops and bring the surface into global alignment [Whe+15].

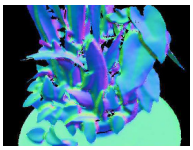
Creating triangular meshes from point clouds can be useful for rendering or further processing. A widely used approach is *Poisson Surface Reconstruction* by Kazhdan *et al.* [KBH06, KH13].



Courtesy of Henry *et al.* [Hen+12]

4.1.4 *Material Reconstruction*

While the geometric reconstruction provides spatial information of the surfaces of the scene and the captured light representation allows to reason about light traveling through it, we lack of the last major component of the scene description, which defines the response of the surface to the light. As discussed



Courtesy of Keller *et al.* [Kel+13]

earlier in Section 2.4, this response is modeled by a BRDF, BSDF or more complex model.

Our own approaches rely on the assumption that the environment surfaces are Lambertian. Hence, there is only one scalar factor to estimate for each surface position $\mathbf{x}$. After gathering the irradiance at $\mathbf{x}$, incident from the upward hemisphere, the diffuse reflectance coefficient is given by the ratio of the observed radiance at $\mathbf{x}$ over the irradiance:

$$f_d(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{L(\mathbf{x}, \boldsymbol{\omega}_o)}{\int_{\mathcal{H}_+} L(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i \, \partial\boldsymbol{\omega}_i} = \frac{L(\mathbf{x})}{E(\mathbf{x})} = \frac{\rho_d}{\pi}.$$

This procedure was applied by Devebec for instance, who used the captured light probe to estimate the diffuse reflectance of objects close to the later augmentations [Deb98] (see Section 4.2).

Despite the fact, that gathering the irradiance for each surface point is not straightforward, the computations for view-dependent BRDFs are much more challenging. Selected approaches are discussed to give an overview of the area.

The material models used in computer graphics today, like the ones in Section 2.4, are simplifications of a more general function that describes the reflectance behavior of an object:

$$\rho(\mathbf{x}_i, \boldsymbol{\omega}_i, t_i, \lambda_i, \mathbf{x}_o, \boldsymbol{\omega}_o, t_o, \lambda_o).$$

It describes the reflectance depending on position $\mathbf{x}_i$, where light of wavelength λ_i hits the surface from direction $\boldsymbol{\omega}_i$ at time t_i and the corresponding outgoing quantities. Note, that the position, direction, wavelength and even the time of a photon leaving the surface do not need to match the corresponding incident values. Hence, it is possible to model complex materials, e.g., ones that show complex subsurface scattering, attenuation or fluorescence. As the positions are embedded in the 2D domain of the surface, the function is 12D and thereby it is not feasible to store the resulting amount of information. Just like for the plenoptic function (4.1), simplifications help to reduce the complexity. Typically, the time and the wavelength are omitted.

When assuming local interactions without subsurface scattering, where $\mathbf{x}_i = \mathbf{x}_o = \mathbf{x}$, we get the Spatially-Varying Bidirectional Reflectance Distribution Functions (SVBRDFs). Since we deal with textured objects most of the time, this is the standard model that is used. It is simply referred to as BRDF, denoted as $f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$.

At this point, it is important to know, that the treatment of all light interactions at the surface of objects is a simplification. In the real world, there are also light interactions inside the object. To capture and model these volumetric effects is challenging. A first step to address them will be discussed next.

LIGHT STAGE MEASUREMENTS The most precise techniques for material reconstruction rely on dome setups containing several cameras and light sources, like shown in Figure 4.5. The object of interest is placed on a turn table in the center of the dome and multiple (often thousands) of images are acquired under different illumination settings. Here, we do not assume local interaction but instead, we assume distant light. Hence, the radiance incident from a certain direction $\boldsymbol{\omega}_i$ is constant for all points of the surface on the object and $\mathbf{x}_i$ can be omitted, which yields the six-dimensional BTF [Dan+99]:

$$\rho_{BTF}(\mathbf{x}_o, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,.$$

Courtesy of Whelan *et al.* [Whe+15]

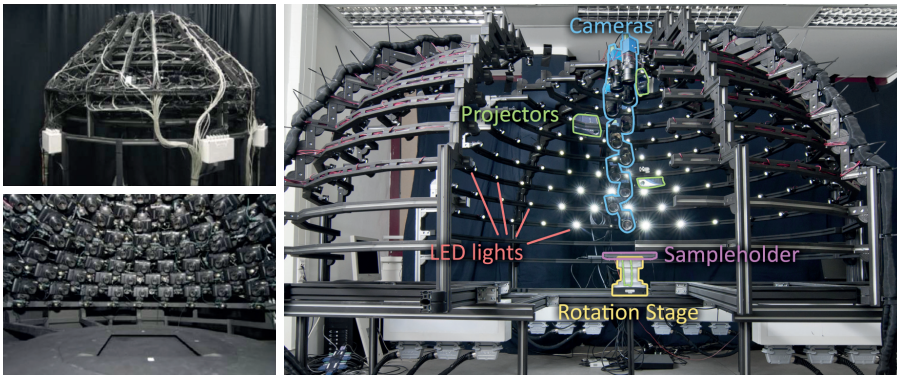


Figure 4.5: Measuring BTFs
Two acquisition systems to capture BTFs of the University of Bonn. Image courtesy of Weinmann *et al.* and Schwartz *et al.* [Wei+16, Sch+14].

Technically, the light sources have a limited non-infinite distance to the scanned object, but in practice, this distance is rather large compared to the size of the measured objects and their geometric details, so the assumption is reasonable [Wei+16].

By using such a dome system, a wide range of effects is covered implicitly. Non-local effects like subsurface scattering, multiple interreflection and refractions, self-shadowing and occlusion are captured together with simple reflections on the surface. Depending on the resolution of the images, very fine details, such as scratches or engravings, that might be visible only from certain directions under specific illumination, get visible in the acquired data.

Note, that due to the setup, the system can also be used to create a geometric reconstruction by using multi-view techniques. Consequently, material and geometry acquisition are usually handled simultaneously.

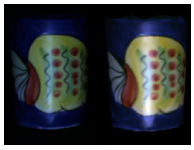
Due to the complex hardware setup, an immense calibration effort is required. The relative position and orientation of all individual components needs to be known. The intrinsic parameters of the cameras have to be estimated and the intensity distribution of the light sources must be known, too. However, the latter can be estimated by capturing mirror spheres in the center of the light dome, while assuming very small distant light source. This in turn requires radiometric calibration of the cameras, reflectance coefficients of the mirror sphere, and if applicable, a calibration of the turn table that used to rotate the object to increase the number of sampled directions.

Courtesy of Schwartz *et al.* [SRK13]

The resolution and the number of acquired images directly influences the size of the data-set. Up to hundreds of gigabytes of data are common even for small objects. Rendering objects using BTFs directly is possible after applying compression schemes. Selecting lower levels of detail allows for more efficient visualization [SRK13]. However, for the application in typical AR scenarios and more classical rendering pipelines, it is desirable to fit measured data into existing analytical models that are easier and faster to evaluate [Mat+03, Gua+16]. An early and very famous example is the BRDF by Ward [War92].

Courtesy of Marschner [Mar98]

Several approaches that follow an similar idea but require less hardware setup have been presented earlier. Marschner for instance measured spatially uniform BRDFs, which can be seen as an early step towards the BTF acquisition [Mar98]



Courtesy of Sato *et al.* [SWI97]

Even earlier, Sato *et al.* reconstructed the shape from images of a rotated object. By also capturing the shading, produced by a known light source, they

were able to estimate diffuse and specular material coefficients. The key idea is to observe each surface point from multiple perspectives to separate the diffuse from the specular component [SWI97].

Lensch *et al.* used a controlled environment to reconstruct spatially varying materials. The system detects different materials of real objects in a robust manner and assigns them to clusters. In a second step, the final BRDF per pixel is described as a linear combination of the basis BRDFs obtained for the clusters. This results in a very flexible and automated system that supports arbitrary combinations of BRDFs to reproduce the measured values [Len+01, Len03].

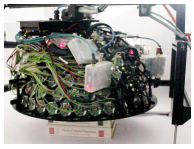
Even though there are portable light stages for BTF acquisition [Hav+17], it is generally not practical for most scenarios to measure all present objects of the scene in advance. An approach to deal with this limitation is to detect materials that are available in a database of measured materials [Liu+10]. As the exact material is usually not present in the database, the material parameters are only approximated. Compared to a correct measurement however, querying is much easier to integrate in a typical workflow. An alternative to measuring material properties is provided by the field of *Inverse Rendering*, covered in the next section.

For more details on measuring BTFs, I refer to recent course notes of Weinmann *et al.* and the article of Schwartz *et al.* [WK15, Wei+16, Sch+14]. The notes also contain detailed descriptions of various other material models and acquisition methods.

4.1.5 *Inverse Rendering*

The field of inverse rendering deals with a fundamental problem in computer graphics and computer vision. Given a geometric representation of the scene and one or more photographs along with corresponding camera parameters, we are interested in material coefficients and a description of the light condition in the scene to be able to reproduce the photographs using a physically-based light transport simulation. Marschner contributed to this area by addressing several subproblems [Mar98].

When the reflectance of the objects is known, the *inverse lighting* problem has to be solved. As the name suggests, the goal is to determine the position, shape and intensity of the emitters in the scene. Assuming distant light, Marschner showed how to solve for incident light by using (regularized) least-squares optimization. He also presented *re-lighting* as an application of inverse rendering, where the illumination in a given images can be altered.



Courtesy of Havran *et al.* [Hav+17]

The second part of his thesis concentrates on *Photographic Texture Measurement*. Here, the lighting in the scene is known and the goal is to capture the spatial variation in the reflectance of an object. The problem to solve is therefore called *inverse reflectometry*. The presented methods are able to estimate diffuse as well as specular reflectance of a previously scanned geometry.

When neither light nor material parameters are known, it gets more complex. However, there are approaches that address this *combined* problem.

Courtesy of Liu *et al.* [Liu+10]

I refer to the survey of Patow and Pueyo that covers important early works, regarding all three categories [PP03]. In the following, only a selected number of works covered in the survey and more recent ones are described to provide an overview. Approaches, that rely on the placement of probes in the scene, are not considered to be inverse rendering techniques in this context.

INVERSE LIGHTING As shown in the classification by PATOW and PUEYO, there are again two subproblems. First, the estimation of the intensity of light sources at a known position and second, the calculation of the position and orientation of the source.

An early example is *Painting with Light* by SCHOENEMAN [Sch+93]. Given a modeled scene containing geometry, materials and the position of light sources, the user roughly draws the desired shading on the geometry. The system then tries to find the intensity and color for each light source that reproduces the drawn shading as good as possible. As lighting is additive, this results in a linear combination with unknown (non-negative) weights.

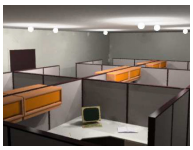
MARSCHNER and GREENBERG presented a similarly descriptive approach [MG97]. Given a photo of an illuminated object, they estimate the incident light as a linear combination of basis images. Each basis image is created by rendering the object with a basis light, where each basis light in turn is defined by a different triangular patch on sphere around the object. Hence, the object is rendered n times and illuminated from different directions. A linear solver can then be used to find the linear combination (with positive weights) to reproduce the input image. The resulting coefficient-vector and the corresponding sphere patches represent the incident light. This approach already assumes distant light and thereby allows to address intensity as well as direction of the incident light.

Courtesy of SCHOENEMAN *et al.* [Sch+93]

Solving the positioning problem and the intensity simultaneously is difficult due to many degrees of freedom. COSTA *et al.* used simulated annealing to generate multiple solution candidates, that the user can choose from [CSF99]. The geometry of the scene, material parameters and some design goals, describing which areas should be illuminated and which not, serve as input. The authors pointed out, that precise knowledge about the properties of the material and the geometry are of extreme importance to achieve acceptable results.

Courtesy of MARSCHNER and GREENBERG [MG97]

In some scenarios it might be possible to start with the estimation of the position, orientation and shape of the lights, before dealing with their intensities. Therefore, highlights and shadows can be analyzed to reason about their cause. Assuming point lights, shadow boundaries and corresponding silhouettes of the shadow casting geometry can be used to set up a linear system and solve for the light position. For area light sources, the umbra and penumbra regions can give clues about the size of the emitter. POULIN and FOURNIER proposed and implemented these ideas in the early nineties [PF92]. In the same paper, they presented an approach to estimate the light direction by picking highlights on an object and by dragging the cursor away from this point, they specified the size of the highlight. For directional and point lights, this allows to estimate the surface roughness. Highly specular material could be used to estimate the extents of the source, but without further knowledge there is an ambiguity at this point.



Courtesy of COSTA *et al.* [CSF99]

More recently, MEI *et al.* also investigated shadows to recover directional distant illumination [MLJ09]. While the approach is limited to simple scenarios with sparse lighting and rough materials, they are able to detect directions that are important for the scenes illumination.

Courtesy of MEI *et al.* [MLJ09]

Machine learning can also be used to address the inverse lighting problem [Man+17]. The idea is to generate synthetic training data for a known model with known diffuse material for many possible vantage points under various illumination conditions. MANDL *et al.* trained Convolutional Neural Networks (CNNs) for each of these vantage points to estimate the environ-

Courtesy of MANDL *et al.* [Man+17]

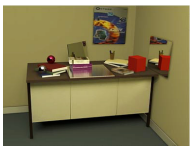
ment illumination in SH basis. During run-time, the camera pose relative to the object is estimated and the corresponding CNN is used to estimate the illumination based on the camera input. By assuming a temporally coherent scene, the light estimation gets more and more complete because of additional data from directions, not estimated before. Training the networks for a simple object however, takes at least one hour using the simplest of the presented strategies. Evaluating the CNN takes about 40 ms on mobile hardware, which results in interactive presentation rates.

INVERSE REFLECTOMETRY In *Inverse Radiosity* we solve for the diffuse reflectance coefficient of each patch. Therefore, it is assumed that the scene is diffuse and that the reflectance as well as the observed radiance is constant for each patch. Given the set of emitting patches, averaged radiance for each patch and known form factors, the coefficients can again be solved for by linear or iterative linear optimization. Many of the early radiosity-based AR techniques rely on this estimation techniques, starting with FOURNIER *et al.* [FGR93]. Because some patches of the scene are not visible in the input images, the average radiance of those patches is not available. The problem can still be approximated, when all patches in the scene are assigned to material groups and when the radiance of at least one patch per group is known – along with the radiance of all emitting patches. In that case, the iterative approach by GROSCH can be used to compute the radiance of hidden surface and to refine reflectance coefficients in an alternating fashion [Gro05b, Gro07].

YU *et al.* also assumed knowledge about material groups to estimate parameters for the Ward BRDF [Yu+99]. While allowing the diffuse albedo to change arbitrarily, they require the specular reflectance and roughness to be constant over predefined groups. The lighting in the scene is measured by one or more light probes, containing the direct sources and the surfaces to estimate. The process – called *Inverse Global Illumination* – iteratively estimates irradiance, radiance and reflectance parameters. Eventually, it yields albedo maps and coefficients for all material groups. Re-renderings of the reconstructed scene data emphasizes the quality of the estimates that can be acquired by the approach.

Courtesy of YU *et al.* [Yu+99]

Another iterative approach with stunning results was presented by BOIVIN and GAGALOWICZ. Given a scene model including the light sources and camera parameters, their approach generates synthetic images with increasingly complex reflectance models [BG01]. Starting with simple Lambertian materials, the synthetic images are compared to the input image. Other models are tested, if the difference is greater than a user-defined threshold. The set of BRDFs to test contains mirrors and specular materials, which can be parametrized to minimize the error compared to input. If all models fail, the approach offers texture maps as fallback solution.



Courtesy of BOIVIN *et al.* [BG01]

With interactive rendering in mind, LOSCOS *et al.* estimated diffuse reflectance coefficients. Several radiance images taken from a fixed camera position but illuminated by known light sources at different positions in the scene serve as input. The estimated reflectance parameters per image are weighted to deal with shadows and specular reflections [Los+99].

Courtesy of LOSCOS *et al.* [Los+99]

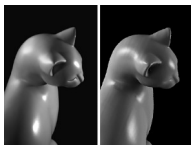
More recently, KÁN and KAUFMANN presented a mobile method for extracting diffuse textures from flat surfaces. After observing the material from many different directions, the minimum pixel value is interpreted as diffuse reflectance coefficient, as specular reflections only increase the reflected radiance [KK15]. Assuming white environment light and proper white balance,

Courtesy of KÁN and KAUFMANN [KK15]

this results in a very easy technique that can be used in many scenarios. The remaining degree of freedom, i.e., the brightness of the environment, can be adjusted by the user if required.

Courtesy of Riviere *et al.* [RPG16]

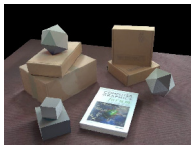
Riviere *et al.* presented two mobile relectometry approaches to acquiring spatially varying isotropic surface reflectance of planar material samples [RPG16]. The first approach makes use of the LED flash light located near the back facing camera of most mobile devices. This allows to capture the light that is scattered back towards the incident direction and thereby the camera. In the other approach, the authors use the display to illuminate the material sample and the front facing camera to capture the reflected (polarized) light. Using polarization filters in front of the camera allows to separate diffuse reflected light (no-polarization) from specular reflected light (partly polarized) [Mül95]. Result of both approaches are textures containing the diffuse and the specular albedo, the roughness and mesostructure details in form of normal maps. All maps can be enhanced in terms of resolution by incorporating close-up observations in an enhancement step.



Courtesy of Ramamoorthi and Hanrahan [RH01a]

Combined Inverse Lighting and Inverse Reflectometry

One of the early works in this area is the one of Ramamoorthi and Hanrahan, who recover lighting and reflectance by deconvolution in SH space [RH01a]. They also showed under which condition this deconvolution can be done robustly.



Courtesy of Gibson *et al.* [GHH01]

Gibson *et al.* addressed the problem from a different perspective. Assuming that only a single image should be augmented, the rest of the scene, i.e., the properties of each individual luminaire and the reflectance of objects outside the field of view, are not required in detail as long as a representation can be found that allows to reproduce the part of the scene that is visible to the camera. Therefore, they used a set of virtual light sources distributed on a sphere around the camera and estimated their intensity distribution. In an iterative process, coefficients of the intensity distributions and material parameters for the visible surfaces (Phong model) are optimized in an alternating manner [GHH01].

Courtesy of Sato *et al.* [SSI03]

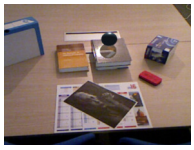
Sato *et al.* estimated the illumination and material properties in a scene based on shadowed and un-shadowed regions in an image. Assuming known geometry, distant lighting, absence of interreflections between occluders and shadowed regions as well as uniform reflectance of the shadowed region, they are able to recover coefficients of a simplified Torrance-Sparrow model and the illumination conditions in an iterative optimization framework. In case the shadowed region has non-uniform reflectance, the parameters can be estimated for Lambertian materials [SSI03].

Courtesy of Grosch [Gro05b]

Grosch projected a single light probe image onto a manually reconstructed geometry [Gro05b, Gro07]. As mentioned earlier in this section, the radiosity of surfaces that are not visible in the spherical image are estimated iteratively. Therefore, it is assumed that material groups are known and that all important light sources are visible in the probe. Eventually, a complete environment mesh with radiosity values for all patches and diffuse reflectance coefficients for the material groups are available for radiosity-based rendering.

Courtesy of Madsen and Lal [ML11]

Later Madsen and Lal presented a technique to estimate the radiance of the sky and the sun in outdoor scenarios [ML11]. By assuming Lambertian materials, they analyzed the shadows of moving objects or persons in a scene with known geometry. This allows to reconstruct the direction and intensity of the sun and the brightness of a hemispherical sky model with uniform



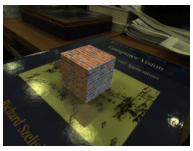
Courtesy of Jiddi *et al.* [JRM16]

radiance. Hence, no special equipment is required other then a calibrated LDR stereo camera system to observe the scene.

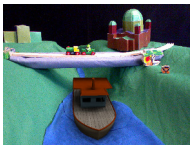
Recently, Jiddi *et al.* analyzed shadows in indoor scenarios using a RGB-D camera. Without assuming a distant environment or distant light, they are estimating diffuse and specular material properties of the local scene as well as the position and intensity of the light source, that responsible for the specular highlights visible in the input images [JRM16]. In a second work, they are focusing on the analysis of shadows [JRM17]. Matching shadowed and unshadowed samples in the area around a shadow casting object, a set of point lights are eventually estimated that reproduce the shadow. These extracted sources are then used for the illumination of virtual objects.

Jachnik *et al.* assumed a static lighting condition and captured a surface light field on a glossy surface by moving a hand-held camera to gather observations from different angles [JND12]. After splitting the light field into diffuse and specular components, the specular part is processed to estimate an environment map containing the incident light. The approach achieved plausible results with no need of special hardware. Therefore, it is well suited for simple AR scenarios.

Gruber *et al.* demonstrated a probe-less approach that displays a visually plausible augmentation based on the input of RGB-D senor [GRS12]. The current light condition is estimated dynamically from observations of many surface points on arbitrary geometry in the scene without any pre-computation steps. A linear system is created from the observations to solve for the incident distant light. Each observation consists of the luminance of the pixel (Lab space), the surface normal and a visibility term. The ambiguity of light and material color is avoided by assuming white light and Lambertian reflectance instead. To estimate the visibility, rays are cast from the sample point into the upward hemisphere and are tested for intersection with the geometric model, reconstructed by using the depth sensor. Samples with very high occlusion are discarded completely as they are not reliable. A binning method is presented that deals with the non-uniform normal distribution of observed real-world surfaces. Eventually, a position independent SH compressed environment map is extracted, which is used for PRT-based differential rendering. Since white light is assumed during the whole process, the residual color information can be interpreted as reflectance coefficients.

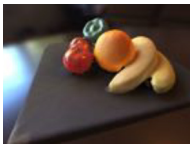


Courtesy of Jachnik *et al.* [JND12]



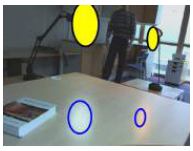
Courtesy of Gruber *et al.* [GRS12]

A method to recover reflectance coefficients and SH illumination from casual RGB-D scans was presented by Richter-Trummer *et al.* [Ric+16]. The scanned objects are segmented into parts with homogenous material properties and the visibility at each vertex is computed by ray tracing. By assuming distant, low-frequency illumination, a factorization of the recorded illuminated surfaces into albedo color and incident light is computed by iteratively solving a linear system that is regularized by using the knowledge about the material segments. Inspired by Jachnik *et al.* [JND12], they then use the information of all vertices of a material segment to estimate the specular reflectance properties for that material. Both, the resulting materials and the extracted illumination can be used for AR rendering by applying the lighting to other objects or by placing the scanned object into scenes with different illumination.



Courtesy of Richter-Trummer *et al.* [Ric+16]

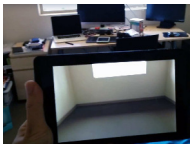
Similar to the idea of Poulin and Fournier, Morgand *et al.* presented an approach to estimate the position and shape of one or multiple light sources based on observed specularities on planar surfaces [MTB16]. By tracking the conic shapes of highlights over multiple frames, they solve for the parameters



Courtesy of Morgand *et al.* [MTB16]

of the corresponding emitters and the material simultaneously. The authors state that they are able to recover color coefficients, an approximated extent of the light source as well as surface roughness with the help of a quadric per highlight. In a follow-up paper, they extended their approach to remove the constraint on planar surfaces [MTB17].

Zhang *et al.* created a system to reconstruct an empty version of a captured room [ZCC16]. After capturing a point cloud using a Google Tango – the same device we use in the approach of Chapter 7 – the authors apply a multi-step offline process to estimate the geometry, materials and light sources in the artificially emptied scene. These steps start with a Poisson surface reconstruction followed by the re-projection of the input radiance onto the mesh. Since all input frames are available during the processing, the unknown exposure for each LDR input frame can be resolved to obtain a globally consistent diffuse appearance of the scene storing HDR radiance information. Assuming a Manhattan-like world, the system identifies major geometric features, i.e., walls, the floor and the ceiling. These features are used to cluster the measurements in material groups. Their Lambertian reflectance is estimated by a constrained non-linear optimization. Simultaneously, the system also solves for light intensities using different light models, i.e., generalized point lights, like spotlights, line-shaped lights and distant light sources for outdoor illumination through windows. Based on the result, the architectural features can be refined to identify doors and other finer details. The offline process takes about 30 to 45 min (excluding the acquisition and the Poisson reconstruction) for simple scenes, because of the Manhattan-world assumption does not allow for complex architecture. The reconstruction is limited to three Lambertian materials, one for each group. The estimated light sources and their intensity distribution are plausible and reliably resemble their real-world counterparts. However, the user needs to specify the number and the type of light sources before.



Courtesy of Zhang *et al.* [ZCC16]

Recently, several approaches to address the problem of inverse rendering under static but unknown illumination have been presented by Dong *et al.* and Xia *et al.* While assuming distant and color-neutral light (i.e., gray on average) as well as no interreflections and no self-occlusions, the approach is still flexible compared to the active illumination methods (see Section 4.1.4), even though the authors used a special device to create a video of the object of interest being rotated. Dong *et al.* propose to iteratively alternate between the recovery of three unknown properties: First, the normal distribution function of an (isotropic) microfacet model, second, the corresponding diffuse and specular coefficients and third, the incident light. To break the ambiguity between surface roughness (sharpness of the BRDF) and blurriness in the incident light, they exploit the observation that the incident light is sparse in the gradient domain, meaning that natural incident light shows few strong discontinuities and is smooth otherwise [Don+14]. Xia *et al.* extended to approach by simultaneously capturing the shape of the object, whereas before, the geometry was assumed to be known. In the extension, they additionally altered between estimation of surface position and normal, the estimation of the incident light, and the estimation of reflectance coefficients [Xia+16].

Courtesy of Dong *et al.* [Don+14]

Courtesy of Xia *et al.* [Xia+16]

A problem related to recovering material properties and incident light from given images is the search for *intrinsic images*. Here, a given image is separated into two components: First, the intrinsic shading, containing the light reflected to the observer, and second, the intrinsic reflection, describing the albedo (diffuse reflectance coefficient) of the surface. Assuming Lambertian

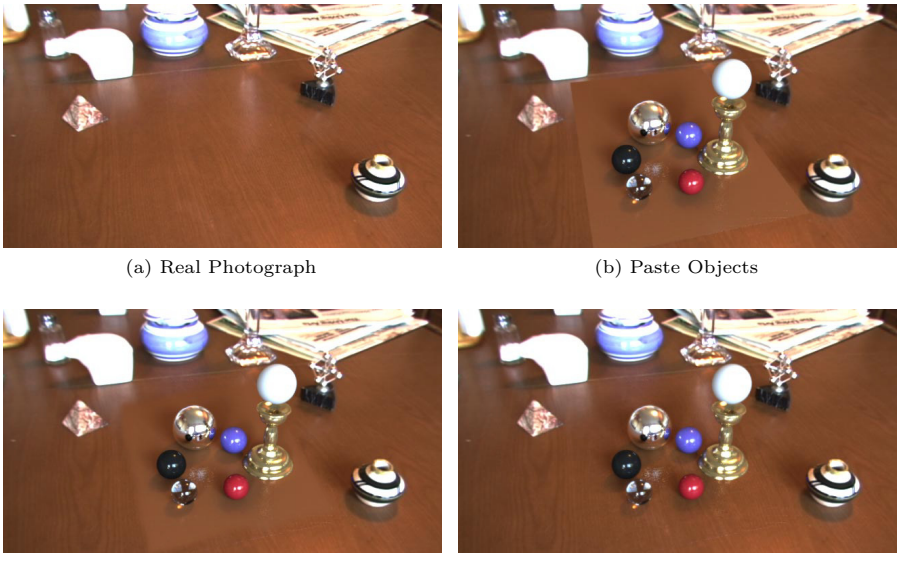
Courtesy of Garces *et al.* [Gar+12]

surfaces, the pixel-wise product of both, results in the given input. Without knowledge about the scenes geometry, this problem is ill-posed and not yet solved completely, just like the inverse rendering. One approach to address the issue is by clustering visible surfaces in Lab color space to segment regions of the same material [Gar+12]. Solving a regularized linear system retrieves the desired intrinsic images. For more details, I refer to the paper of Garces *et al.* as it contains references and comparisons with several other approaches in the field [Gar+12]. An introduction as well as ground truth datasets are provided by Grosse *et al.* [Gro+09].

4.2 Differential Rendering

Differential Rendering is one of the most used concepts in the discipline of AR rendering. The important idea is as simple as elegant and goes back to Alan Fournier *et al.* [FGR93] and Paul Debevec [Deb98]. Since then, nearly all works addressing coherent AR rendering make use of this basic principle and so do we for the presented approaches in Chapter 5 and 7.

First, let us explore our options to superimpose a given camera image by virtual objects. From the previous sections, we know about the real-world geometry in the scene (see Section 4.1.3) and we can estimate the camera pose as well as the intrinsic parameters, used while capturing the input image (see Section 3.3 and 3.2). This allows to place virtual objects with correct perspective and relative size at any position in the mutual world coordinate system. Occlusions between virtual and real objects can also be handled by filling the z-Buffer with the reconstructed geometry before rendering virtual



(a) Real Photograph (b) Paste Objects

(c) Blend Reconstruction (d) Differential Rendering

Figure 4.6: Combining a Real and a Virtual Scene
Different options to superimpose the real scene visible in the input photograph (a). First, the reconstruction of the real object that supports the virtual ones is rendered and pasted onto the background (b). Second, instead of simply pasting the rendering, it can be blended smoothly (c). Third, differential rendering is used to only consider the influence of virtual objects (d). Image courtesy *(a, b, d)* of Debevec [Deb98].

objects. Furthermore, we use the estimated light condition to compute the direct illumination of the virtual object. At this point, we are lacking one very important visual cue: shadows. If physical coherence can be neglected, simple *blob shadows* or *fake shadows*, projected on an invisible plane below the object [Bli88], can provide the cue and improve the perceived location of the virtual object [WFG92, Hu+00]. Without any shadow, the object seems to be floating, as visualized in Figure 3.1. An easy way to incorporate such simple shadows is to reduce the brightness of the background image, which can remain unnoticed by the user, especially when the real emitters are white. By considering the reconstructed material and the extracted light sources, the final color of shadows can be computed more accurate. Rendering all reconstructed objects that get influenced by the virtual ones, e.g., by their shadows, and replacing the corresponding parts in the input photograph provides the necessary cues (see Figure 4.6b). However, the boundaries between the real and the virtual elements are probably visible because of the approximations made during the reconstruction. Blending between the rendered image and the background improves the result (Figure 4.6c), but boundaries are probably still noticeable. Now, there is differential rendering, which requires exactly the same input as the last two options, but it achieves visually more convincing results, as real and virtual objects blend more or less seamlessly. The following deliberations are based on the formulation of Debevec [Deb98].

Computing two light transport simulations in the reconstructed scene is the key idea behind differential rendering. For the first simulation, we render an image, denoted as L_r, that contains the reconstructed scene only. The goal here is to resemble the input photograph L_c as good as possible. Because of inaccuracies in the reconstruction, there is a remaining error e:

$$e = L_r - L_c. \tag{4.2}$$

The second simulation includes all reconstructed and virtual elements resulting in the image L_{r+v}. In our examples, we consider virtual objects only, but additional virtual light sources are possible, too. Even removing light sources or altering materials is supported without conceptual changes. However, assuming that the error between L_{r+v} and a perfectly augmented image equals the error of the first simulation, we can compute our final image L_f, by:

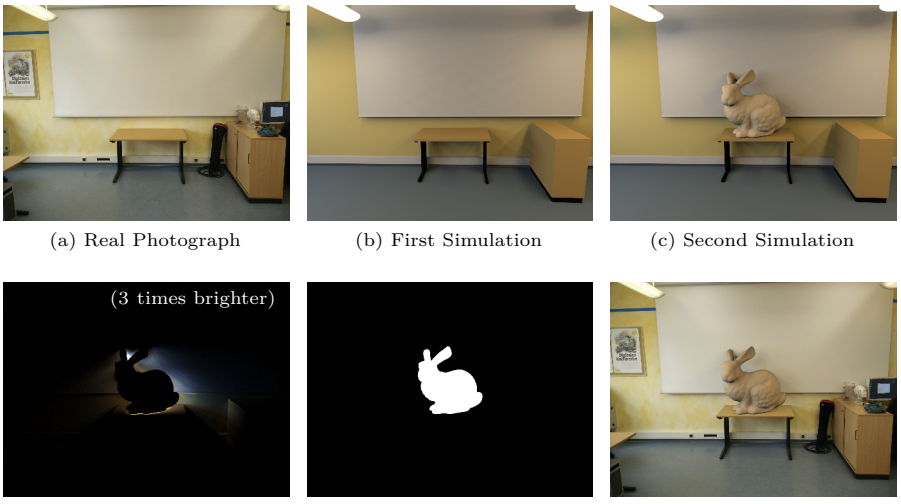
$$L_f = L_{r+v} - e. \tag{4.3}$$

Substituting Equation (4.2) yields the differential formulation:

Equation for Differential Rendering

$$\begin{aligned} & L_f = L_{r+v} - (L_r - L_c) \\ \Rightarrow \quad & L_f = L_c + (L_{r+v} - L_r) \\ \Rightarrow \quad & L_f = L_c + \Delta L, \end{aligned} \tag{4.4}$$

where $\Delta L = L_{r+v} - L_r$ can be interpreted as the influence of the virtual elements on the real environment. Since the assumption of equal error in both simulations makes no sense for pixels that show the virtual object, a mask is used to switch between two cases: If the pixel contains a virtual object, we simply show the virtual object, i.e., L_{r+v}. Otherwise, we add ΔL to the camera image. All steps are visualized in Figure 4.7. Note, that the difference image in Figure 4.7d contains absolute values for illustration purposes. In the shadows of virtual objects, the difference is negative, which results in a darkening of the final augmented image. When the light transport simulation considers further global illumination feature, like indirect lighting, the difference image



(a) Real Photograph (b) First Simulation (c) Second Simulation

(d) Absolute Difference (e) Virtual Object Mask (f) Augmented Photograph

Figure 4.7: Differential Rendering
Differential Rendering allows to augment a real photograph (a) by performing two lighting simulations of the reconstructed environment: one without (b) and one with virtual objects (c). The difference between both simulations (d) is added to the real photograph (a) resulting in the augmented image (f). Pixels showing a virtual object are treated separately based on a mask (e) and replace the background completely.

can also contain positive values. Adding virtual light sources would also result in a positive impact.

Thanks to the fact that light can be added linearly, we can simply add ΔL to the input camera image without introducing any further error. However, Debevec also suggested to use the ratio instead of the distance between the two simulations. In case of uncertain reconstructions, this can yield to a more robust solution but introduces a systematic error. In case of pixels showing no virtual object, the final color would be computed as follows:

$$L_f = L_c \, (L_{r+v} / L_r).$$

While the general idea of differential rendering already appeared in the work of Fournier *et al.*, the formulation of Debevec contains several new aspects. One of them is the partitioning of the scene into three components as visualized in Figure 4.8. Because of the cumbersome reconstruction that is also prone to errors and the observation, that the notable influence of virtual objects is usually limited to nearby geometry, it makes sense to reduce the effort and limit detailed reconstructions to the *local scene*. The term *local* is not meant literally, instead, all real-world objects that are influenced by the virtual object, such as distant mirrors, are included. Virtual objects form the second component and the remaining part of the scene is denoted as the *distant scene*, which is represented by a purely light-based model, e.g., a light probe captured in the vicinity of the virtual object. Debevec also proposed to project the light probe onto a coarse model of the real scene, e.g., a simple box with the approximated extents of the room. Due to the assumption that virtual objects do not influence the distant scene, the light transport towards surfaces in the distant scene is zero and can be neglected. Hence, we do not need any reflectance model for these surfaces. The reconstruction of the local

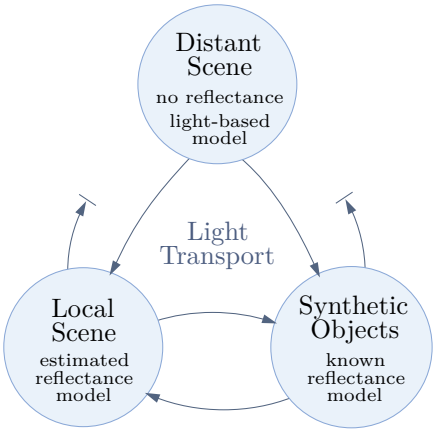


Figure 4.8: Scene Partitioning
The scene is subdivided into three components: the distant scene, the local scene and the virtual scene. There is no light transport computed towards the distance scene, i.e., it is assumed that the virtual object influences the local scene only. Image courtesy of Debevec [Deb98].

scene is often reduced to a minimum. Therefore, only a planar surface that supports the added virtual object can be sufficient for simple scenarios. Often, the reflectance of that surface is assumed to be Lambertian, which further simplifies the augmentation [SSI99, GM00, Kor+07].

The concept, as presented here, still has some disadvantages. First, reflections in glossy or specular virtual objects show the reconstructed scene rather than the real-world. Because of the often very coarse approximated local scene, for instance missing texture details can be easily noticed by the user. Second, the partitioning of the scene is not working in every case. A specular or mirror-like virtual object could reflect strong light far into the scene or the virtual object could emit light. In both cases all surfaces have to be considered *local* or the missing light paths towards the distance scene cause perceivable errors. Another issue is efficiency. As two light transport simulation have to be conducted and the difference between both results is usually small and concentrated in local regions, a lot of computing resources are wasted. A way to address this issue is to conduct one simulation only, that computes the difference directly, as shown in *differential photon mapping* by Grosch [Gro05a]. Here, the tracing of photons, blocked by virtual objects, is continued with negative energy. Storing this *anti-radiance* at the positions, where the non-blocked photon would have ended, yields shadows in the subsequent radiance estimation step. This however, requires a special renderer, while Debevec's original approach comes down to two standard light simulations and a simple post-effect. When aiming for real-time performance, such a dedicated renderer is absolutely necessary as seen in the next section, which contains a selection of existing interactive coherent rendering techniques for AR.

4.3 Interactive Coherent Rendering


Courtesy of Grosch [Gro05a]

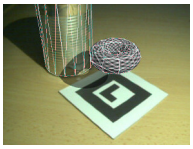
The focus of this thesis is on consistent rendering in interactive and mobile AR scenarios. Hence, consistent rendering, especially in terms of photometric registration (see Section 3.1), is of major interest. To provide an overview of the related works, this section will be used to present selected approaches.

The classification of Jacobs and Loscos serves as basis to structure the methods [JL06]. Since *Inverse Rendering* was covered in Section 4.1.5, only the other three categories related to image synthesis rather than reconstruction are considered here. Approaches with *Local Common Illumination* compute the appearance of virtual objects under a measured or guessed light condition in the scene. The real and virtual objects might cast shadows onto each other, even though this technically is a global phenomena. However, there is no indirect light reflected from virtual objects onto real surface in this group. In case there is at least one such bounce of indirect light in both direction – real to virtual and virtual to real, the approach is assigned to the class of methods with *Global Common Illumination*. The last group, *Common Illumination and Relighting*, focuses on the altering of real-world scene elements, i.e., materials, light sources or geometry. Adding virtual light sources that illuminate the real-world surfaces is considered a feature of this group, too.

Furthermore, approaches that address *Camera Simulation*, the third aspect of *Visual Coherence*, are presented in Section 4.3.4. These approaches constitute the related work of parts of the method presented in Chapter 7.

No claim is made that this overview is a complete survey. Instead, important or unique interactive approaches that have been presented in the past and mobile approaches of the recent years, have been selected, as they are related to the own methods or inspired them in some way.

4.3.1 *Local Common Illumination*



Courtesy of Haller *et al.* [HDH03]

Courtesy of Jacobs *et al.* [Jac+05]

Courtesy of Gibson and Murta [GM00]

Courtesy of Gibson *et al.* [Gib+03]

Predefined light sources Several approaches assume the light sources to be given and simply define the existing illumination manually, e.g., Haller *et al.* [HDH03], who implemented shadow volumes [Cro77] for the ARToolKit[20].

Jacobs *et al.* proposed a technique that produces consistent shadows, i.e., same brightness and no double shadowing. The approach is also based on shadow volumes, but limited to scenarios with one light source that is known approximately. Unlike in differential rendering, the goal is to make shadows appear consistent rather than evaluating a physically-based model. Therefore, they detect real shadows, create a protection mask to avoid double shadowing and third, generate shadows that include virtual objects while their brightness is scaled to match the real ones [Jac+05].

Static light probes Using a single light probe image – e.g., captured using a mirror sphere – leads to the assumption of distant and constant illumination. This is very common in the field of IBL but also in AR.

A first interactive approach for AR with consistent illumination was presented by Gibson and Murta, which also was one of the first GPU-based techniques [GM00]. As suggested by Debevec and Malik [DM97], they acquire a HDR light probe. Pre-filtering the probe (see Section 2.7.4) yields a diffuse irradiance map as well as specular maps for materials of varying roughness. The authors also extracted a set of discrete directional light sources from the light probe and used these sources to cast overlapping shadows using hardware shadow mapping. The overlapping of four to eight shadows result in acceptable (soft) shadows.

20 Daqri. *ARToolKit*. Project Website, 2017.

Revisiting the idea of overlapping shadows, Gibson *et al.* [Gib+03] presented another important work that is closely related to the earlier radiosity-based method. Given a patch-based reconstruction of the scene geometry, they classified the patches into two (non-disjoint) categories: source and receivers. While the N brightest patches, containing 70 % of the total radiance in the scene, are considered important sources of light. In an pre-processing step, the radiance transfer between source and receivers are computed by form factors and visibility estimation. The line-space that is spanned by connections between source and receiver patches is represented by a shaft-hierarchy. By intersecting the bounding box of virtual objects, blocked light paths can be identified quickly and are stored in a list. Then, the list is iterated for differential rendering, while applying shadow mapping to evaluate visibility more detailed. Multiple overlapping hard shadows again appear as soft shadows. By selecting higher levels in the hierarchy, a quality vs. performance trade-off allows to reach real-time performance. The synthetic objects are shaded using an irradiance volume, which stores SH compressed low frequency light, and a dynamically updated environment map for specular reflections. Hence, near-field illumination was taken into account by this approach already. The system was mainly limited to Lambertian reflectance but allowed to add a specular layer using the environment map for reflections of (distant) lights and objects.

Courtesy of Agusanto *et al.* [Agu+03]

Agusanto *et al.* capture a static HDR light probe and store it in an environment map. After pre-filtering, they where able to show diffuse and glossy objects at interactive frame rates. Shadows were only included in their multi-pass rendering system, but not in their implementation for the ARToolKit at that time [Agu+03]. However, the authors already aimed for photorealistic rendering and based their illumination computation on the rendering equation (2.12).

Courtesy of Pessoa *et al.* [Pes+10]

Years later, Pessoa *et al.* projected a static light probe onto a box and rendered new environments for individual virtual objects [Pes+10]. They use image-based lighting with the Lafortune BRDF extended by a Fresnel term and render virtual objects with various materials. For shadows, they manually specified a set of light sources and computed shadow maps.

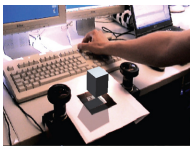
Courtesy of Kanbara and Yokoya [KY02]

DYNAMIC LIGHT PROBES Continuously updating the light probe allows for dynamic illumination from a distant scene. A very common setup involves a second camera equipped with a fish-eye lens to acquire the dynamic light probe.

Another early method was presented by Kanbara and Yokoya [KY02]. Using their probe that combines a 2D marker for tracking and a mirror ball to acquire the environment light condition, they use standard forward shading and shadows to render a virtual teapot. Technically, this simple methods can also handle dynamically changing light conditions and operates at 15 Hz.

Courtesy of Havran *et al.* [Hav+05b]

Designed for temporally varying illumination, Havran *et al.* introduced a sampling approach for images of a HDR video camera [Hav+05b]. The HDR environment map is decomposed into a set of directional light sources by applying importance sampling. Incorporating stratified sampling yields good temporal coherence and spatial distribution of the extracted directional lights. Discarding irrelevant light sources and clustering important ones, reduces the computation effort for the shadow map generation and the rendering. This enables interactive performance or progressive refinement.



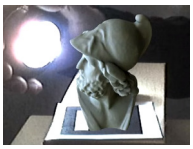
Courtesy of Korn *et al.* [Kor+07]

Korn *et al.* used two such HDR video camera [Kor+07]. After a proper intrinsic and extrinsic calibration, the position of a few real point light sources are calculated by epipolar geometry and tracked over time. These light sources are then used to shade virtual objects that are inserted into the live images of an additional webcam. An ambient term is added to compensate for the remaining illumination of the scene.

Grosch *et al.* presented an approach to compute near-field illumination for augmentations under daylight in a real room [GEM07]. Light, that is entering the room through a window, is captured by a HDR video camera equipped with a fish-eye lens. The authors separated direct from indirect light transport. Therefore, the direct light is handled by sampling the camera image, similar to Havran *et al.* To deal with dynamic indirect light, the authors proposed to use an irradiance volume that stores only indirect light traveling inside the room. This is realized by subdividing the fish-eye image into N patches. For each region, a light simulation is conducted, where the current patch is used as light source of unit brightness and all other patches are black. Measuring the simulated irradiance at the grid cells of the irradiance volume yields a *basis irradiance volume* that is compressed using SH. During runtime, the SH coefficients of the basis volumes are accumulated, weighted by the current average radiance in the corresponding fish-eye patch. A PRT variant is then used to add indirect light to the shading of the virtual objects inside the room. The approach is limited to Lambertian materials because of PRT.

Courtesy of Grosch *et al.* [GEM07]

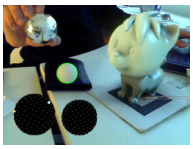
Franke and Jung presented a straightforward PRT solution in which live spherical images, provided by a fish-eye camera, are projected into SH basis [FJ08b]. Even though the idea is simple and does not provide shadows, it results in a reasonable shading at real-time frame rates. Depending on the transfer function stored per vertex, e.g., indirect diffuse illumination or sub-surface scattering can be incorporated with no additional run-time cost. In another work, they incorporated soft shadows from direct light sources, extracted in a pre-process, using filtered shadow maps. They also substituted PRT by AO in order to handle non-rigid transformations of virtual objects [FJ08a].



Courtesy of Franke and Jung [FJ08b]

Courtesy of Franke and Jung [FJ08a]

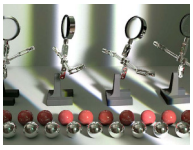
Miika Aittala used only a diffuse sphere to recover low frequency incident illumination but also supported the moving planar marker of Pilet *et al.* as an alternative probe [Ait10]. In contrast to a light probe, that stores incident radiance from all directions, the authors used eight directional light sources with fixed orientation and solved a linear system to fit intensities for each source to resemble the observed radiance on the probe. The residual error is projected into SH basis and applied on top using PRT. He computed soft shadows for each of the directional lights.



Courtesy of Miika Aittala [Ait10]

Assuming a single dominant light direction, Nowrouzezahrai *et al.* introduced a real-time light factorization method that allows soft and hard virtual shadows [Now+11]. A continuously visible mirror ball provides a dynamic light probe, which is projected into SH basis and split into two terms: One term that corresponds to a sun-like directional source, mainly responsible for the scenes illumination, and a global term that contains the remaining incident light from the rest of the environment. Using (rotated) *Zonal Harmonics*, the symmetric part of the SH basis ($m = 0$), enables SH-based illumination for animated objects. The authors propose to use a sphere-based approximation of the virtual object to produce soft shadows from the global part of the environment light. The soft shadows provide an increased perceptual consistency in addition to a shadow map for the directional light.

Courtesy of Nowrouzezahrai *et al.* [Now+11]

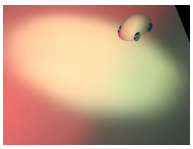


Courtesy of Unger *et al.* [UGY07]

LIGHT FIELDS Measuring the light condition at multiple locations allows for spatially varying illumination without the distant scene assumption. Rendering such a light field consisting of many light probes at different locations is all about selecting the correct probe and sampling the correct direction. The sampling is not straightforward, since mirror spheres have a certain radius, which is usually ignored for standard IBL techniques relying on a single probe [UGY07]. The group around JONAS UNGER worked not only on capturing light fields, but also on the corresponding rendering techniques. One approach involves to back-projection from an auxiliary geometry above the local scene to select the probes for interpolation [Ung+03]. Because of the immense amount of data and classical IBL-based approaches that serve as foundation, those techniques are not suited for interactive applications, as rendering takes several hours.

Courtesy of UNGER *et al.* [Ung+03]

The compressed light fields of LÖW *et al.* achieves real-time performance for mostly low-frequency illumination [Löw+09]. Even though local occlusions are not handled, which results in a lack of shadows, they show, that re-sampling and efficiently storing the light fields is a promising direction for the future.



Courtesy of LÖW *et al.* [Löw+09]

PROBE-LESS ILLUMINATION ESTIMATION MEILLAND *et al.* presented an image-based representation of the scene [MBC13] (also see Section 4.1.2). Even though it should be possible to directly use this light field for differential rendering, the authors created virtual light probes at the centers of the synthetic objects and extract a few discrete light sources for shadow computations. Because their reconstructed model also contains position data, point lights or even area light could be used instead of the common directional lights. However, differential soft shadows are added by applying variance shadow maps [DL06].

Courtesy of MEILLAND *et al.* [MBC13]

The probe-less approach of GRUBER *et al.* [GRS12] also uses SH and PRT for rendering augmented images. They are able to present plausible shadows, cast by virtual objects. Therefore, the transfer function (visibility) per pixel in the augmented scene is estimated via ray tracing in the geometric model, acquired by the RGB-D sensor. While allowing for dynamic illumination, geometry and camera movement without any pre-processing or probes, the authors made a step towards the general applicability of AR. The SH-based light representation used for rendering assumes distant white light. Later, the method was extended by an adaptive sampling and caching strategy [Gru+14], that improves the performance or allows for larger sample counts and thereby higher quality, compared to computing visibility for every n^{th} stratified sample. The performance was further improved by realizing an image-space variant of their technique [GVS15] (see next section).

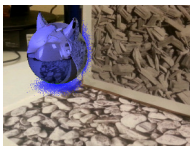
Courtesy of GRUBER *et al.* [GRS12]

Courtesy of GRUBER *et al.* [Gru+14]

4.3.2 *Global Common Illumination*

The idea of differential rendering has been applied to several interactive global illumination methods.

STATIC LIGHT PROBES CSONGEI *et al.* presented a progressive path tracing solution for AR on mobile phones based on a pre-recorded environment map. Here, the illumination is simulated on a stationary PC and streamed to the mobile device. Besides the low quality of the light transport simulation, there is a trade-off between the quality of the steaming, bandwidth and latency [Cso+14].

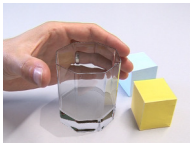


Courtesy of CSONGEI *et al.* [Cso+14]

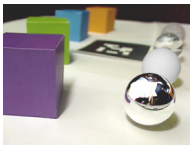
DYNAMIC LIGHT PROBES KNECHT *et al.* applied instant radiosity (see Section 2.7.2) to AR in order to simulate mutual effects of virtual and real objects [Kne+10]. A camera with fish-eye lens continuously captures the lighting in the scene, including a moving real flashlight, the main light source, for which a RSM is created. VPLs are (recursively) generated from the RSM as well as from samples drawn in the fish-eye image. The two light transport simulations – for differential rendering – are computed in one rendering step and imperfect shadow maps [Rit+08] are used for visibility tests. The approach occasionally suffers from double shadowing artifacts because the RSM and the shadow maps contain information only for the first visible surface. However, the technique is able to produce plausible color bleeding from real to virtual objects and vice versa. Additionally, a virtual light source can be used instead of the flash light, which adds a relighting feature.

Courtesy of KNECHT *et al.* [Kne+10]

Progressive path tracing and thereby a global illumination technique, was applied to AR by KÁN and KAUFMANN [KK12a]. They showed interactive augmentations containing high-quality specular effects such as refractions and caustics. NVIDIA's real-time ray-tracing engine OPTIX[21] allowed to port the classically offline methods, namely photon mapping and path tracing, to the GPU. In the same year, the authors added physically-based depth of field for AR to their framework and thereby further increased the quality of the results [KK12b, KK13b]. While the quality of these previous approaches improved slowly with the number of progressive iterations, the authors improved the convergence speed by applying irradiance caching to exploit the spatial coherence of the indirect illumination [KK13a]. The authors suggest a re-projection technique to splat the irradiance into the frame buffer instead of more expensive tracing (or even final gathering). The re-projection allows to handle depth of field, anti-aliasing and refractions even though irradiance spatting is applied.

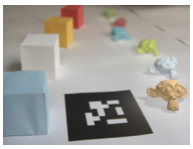


Courtesy of KÁN and KAUFMANN [KK12a]

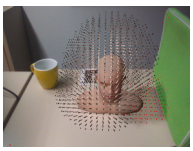


Courtesy of KÁN and KAUFMANN [KK13b]

TOBIAS FRANKE presented a differential version of LPV (see Section 2.7.7) to spread the changes in illumination that are caused by the presence of added virtual objects [Fra13a, Fra13b]. Therefore, an irradiance volume around the virtual object, approximately twice as large as the object, is allocated. Two RSMs are computed for each reconstructed light source: RSM_r, containing only real objects and RSM_{r+v}, which also includes the virtual object. Then, secondary lights, created from RSM_{r+v}, are injected into the volume, followed by negative secondary lights, created from RSM_r. To also incorporate direct shadows, the direct light is injected into the volume similarly. The resulting difference at each cell is positive in areas with strong indirect light and negative in regions of virtual shadows. For rendering, the diffuse reflectance coefficient of the reconstructed geometry is multiplied by the queried value from the Delta Light Propagation Volume (DLPV) and added to the camera image.



Courtesy of KÁN and KAUFMANN [KK13a]



Courtesy of FRANKE [Fra13a]

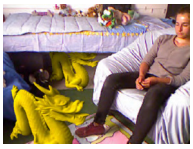
FRANKE also introduced Delta Voxel Cone Tracing (DVCT) [Fra14a, Fra14b]. The differential version of VCT, presented by CRASSIN *et al.* (see Section 2.7.8), inherited the benefits from the original. Hence, it is able to produce diffuse and specular reflections in completely dynamic scenes without pre-processing. He adapted the idea of injecting positive and negative VPLs from DLPVs and created a *delta-volume*, containing radiance and anti-radiance caused by the introduction of the virtual object. The influence on the real geometry is estimated by casting one shadow cone towards the light source for sampling the blocked direct illumination, and a final gathering step. For the latter, nine cones are traced in the delta-volume to gather data for diffuse reflection and

Courtesy of FRANKE [Fra14a]

21 NVIDIA. *OptiX™ Ray Tracing Engine*. Developer Website, 2017.

one additional cone for specular highlights. A third volume, besides the ones for opacity and the delta-volume, stores the first bounce indirect illumination for the augmented scene. For rendering the synthetic objects with GI effects, this volume is traced respectively. Similar to the original paper, this is one of the fastest rasterization-based methods, that offers specular interreflections. It is faster than the approach of KÁN and KAUFMANN, but the quality depends on the resolution of the volume. For low resolutions or highly glossy materials, the voxels become visible in specular reflections. Furthermore, reflections show only the directly illuminated voxels, which is also noticeable in these cases. Like in VCT, using the approach will also show light-bleeding through thin geometry or when tracing on coarser levels of the hierarchy. Another limitation is the memory consumption, as the approach is based on a dense volume.

PROBE-LESS ILLUMINATION ESTIMATION GRUBER *et al.* [GVS15] further improved their earlier method (see Page 97). Therefore, they increased the quality of the scene reconstruction by merging dynamic and static geometry information, followed by a smoothing step. This allows to handle moving real-world objects better than with conventional RGB-D sensor data. The other important contribution of this work is the replacement of the purely ray tracing-based estimation of the visibility for surface points in the current field of view. A variant of SSDO [RGS09] is used to estimate the directional visibility in screen-space. Different to SSDO, they do not gather incident radiance from the environment, but rather project the visibility into SH basis, which is then used as input for the earlier presented PRT-based differential rendering. The screen-space approach supports indirect light from virtual to real objects too, but inherits the limitations of the former approach. Hence, the assumed diffuse reflectance of real surfaces is not estimated correctly, as light is defined to be white. Light is also assumed to be distant and not handled in spatially varying manner.



Courtesy of GRUBER *et al.* [GVS15]

4.3.3 *Common Illumination and Relighting*

PREDEFINED LIGHT SOURCES A first method to augment live images based on geometry, that is captured by a RGB-D camera, was presented by LENSING and BROLL [LB12]. Without a complete scene reconstruction and with no information about the light condition, they added a virtual (spot) light to illuminate virtual objects. After creating a RSM for each light, VPLs are generated as sources of indirect illumination from synthetic onto real geometry and vice versa. Besides the application of guided image filters to improve the normal estimation from RGB-D data, a multi-resolution algorithm for splatting indirect light was presented. The screen-space method is based on the work of NICHOLS and WYMAN [NW09] and allowed for a high number of VPLs and thereby improved temporal coherence at real-time rates. In another work, LENSING and BROLL presented their *LightSkin* technique, which was also applied to Mixed Reality (MR). Here, they reduce the number of points to compute shading for, to a set of samples on the surface of the object. An interpolation technique is then used to estimate the shading of all other surface points, which allows for real-time glossy reflections [LB13].

Courtesy of LENSING and BROLL [LB12]

Courtesy of LENSING and BROLL [LB13]

STATIC LIGHT PROBES GROSCH showed how the technique of GIBSON *et al.* [Gib+03] can be further improved for the augmentation in a

Courtesy of GROSCH [Gro05b]

panoramic image viewer [Gro05b]. The PANOAR-tool additionally allowed to change materials of reconstructed surfaces and to remove real objects.

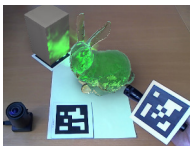
DYNAMIC LIGHT PROBES Differential instant radiosity by KNECHT *et al.* allowed to use a virtual spotlight to illuminate real and synthetic objects [Kne+10]. Since, relighting is not focus of that paper, it was discussed before (see Page 98) and is only mentioned here.

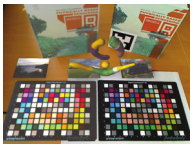
In a later work of KNECHT *et al.*, they used MICROSOFT's KINECT sensor to avoid the manual reconstruction process. They also addressed the double shadowing problems by using two RSMs and two sets of imperfect shadow maps [Kne+12]. In another extension, they integrated reflective and refractive objects into the framework [Kne+13]. The new approach used splatting to produce caustics that also emit indirect light and thereby added another type of light path, that increases the plausibility of the augmentations.

4.3.4 *Camera Simulation*

Courtesy of KNECHT *et al.* [Kne+10]


Courtesy of KNECHT *et al.* [Kne+12]


Courtesy of KNECHT *et al.* [Kne+13]


Courtesy of KNECHT *et al.* [Kne+11]


Courtesy of KÁN *et al.* [Kán15]

As mentioned earlier, several aspects related to the camera influence the image that will eventually be augmented by virtual objects. These aspects include artifacts like lens distortion, blur, noise, vignetting, chromatic aberrations and artifacts caused by the Bayer mask. For more details on these issues, I again refer to book of SCHMALSTIEG and HÖLLERER [SH16].

One important aspect for this thesis however, are the artifacts caused by a missing radiometric calibration (see Section 3.4). Since such a calibration is not feasible for every consumer device [SH16] and because of the lack of detailed manual control during the image acquisition, another kind of color compensation must be included, when recording real-world radiance values using such mobile device.

For most mobile devices, exposure time and white balance of the camera change automatically without manual control. To address this issue, KNECHT *et al.* adaptively map the colors of virtual objects to the colors present in the current camera image [Kne+11]. The general idea is to create a mapping between the camera image L_c and the partial solution L_r of the differential rendering (see Section 4.2). Applying the same mapping to virtual objects yields coherent colors. The authors suggest to fill gaps in the map by simple heuristics, that involve swapping color channels. Based on the evaluated examples, they state that this improves the result. However, to compute L_r, a scene reconstruction containing material properties is required. Hence, if the goal is to reconstruct materials and to estimate the light condition of the scene, the approach cannot be used. In Chapter 7, we present a color compensation that can be used during the reconstruction and works without a heuristic like the one of KNECHT *et al.* The approach mainly focuses on devices with no control of the acquisition parameters and no meta-data per image. The video preview mode that is used for AR applications is subject to these restrictions [Kán15].

In contrast, if the photo mode is used, more detailed options and meta-data, stored along with the images, allow for more sophisticated workflows, known from professional cameras. KÁN showed that stitching an HDR environment map from images captured using a mobile device is possible if the exposure of the camera is known [Kán15]. Like KÁN, e.g., MEILLAND *et al.* [MBC13] assume that all camera parameters can be fixed. Allowing the shutter speed

to be adjusted automatically results in a single degree of freedom, that can easily be estimated or found in the meta-data of the images.

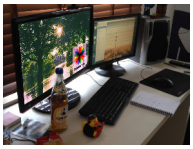
Recently, exposure correction methods were used for the correction of recorded color values of a KINECT camera by RICHTER-TRUMMER *et al.* (see Page 88) and for the color values of a tablet camera by ZHANG *et al.* (see Page 89) [Ric+16, ZCC16]. These solutions search for a global optimum of a given set of images.

A common related problem, especially in the area of advertisement, is the placement of objects into an image, that have been cut out of another images. A simple solution consists in the alignment of the color spaces based on image statistics, e.g., by aligning the histogram or by aligning the mean and average color distributions of the images as suggested by REINHARD *et al.* [Rei+01].

LALONDE and EFROS investigated color statistics in large image collections to eventually decide if the color palette, used in an image, appears naturally and thereby to distinguish coherent from incoherent looking images. They suggest to combine these global statistics with the local approach of REINHARD *et al.* to create compositions with improved perceived color consistency [LE07]. In the context of his bachelor's thesis, TIM GERRITS investigated if an automatic white balance can be used to align the illumination of a given object with the light condition in the background scene [Ger14]. The results show that there is no superior method that solves this problem. In some cases, adjusting the white balance and in other cases, approaches, like the one of REINHARD *et al.* or follow-up works, like to one by CIAO and MA [XM06], are preferable.

Courtesy of REINHARD *et al.* [Rei+01]

Courtesy of LALONDE *et al.* [LE07]



Courtesy of TIM GERRITS [Ger14]

Courtesy of XIAO and MA [XM06]

4.4 SUMMARY

Environment reconstruction based on images or videos is a challenging task. Even when breaking it down to geometry, light and material estimation, none of the subproblems is solved yet. At least not to a degree, that allows for photorealistic AR applications in unknown environments. Besides the required computation time, most of the processes still rely on special hardware or a serious amount of user interaction before being able to start rendering virtual objects.

For representing the environment light condition, light probes are the most common solution. To address dynamic settings, HDR cameras equipped with fish-eye lenses are used as an alternative. Light probes as well as the light information extracted by most of the inverse rendering techniques store directional radiance only. Hence, they are not able to represent spatially varying illumination nor near-field illumination.

In the area of geometry reconstruction, approaches based on depth-sensing devices are currently the most dominant ones. Approaches like *Kinect Fusion* enable fast acquisition of dense voxel-based geometry. However, the size of the acquired dataset grows rapidly for larger environments, especially when storing other surface information, like material parameters. Due to the limited field of view of the cameras, dynamic geometry cannot be handled completely at the moment. Even if the moving objects are in the field of view, it can be difficult to update the scene representation without delays, as there are global dependencies, caused by the interaction of light while traveling through the scene. However, most approaches assume a completely static environment.

Most approaches assume Lambertian materials, as the view-independent diffuse reflectance can be estimated rather easily, compared to the view-dependent properties. Unfortunately, many real-world surfaces, especially the man-made, are glossy or even specular. To acquire the reflectance behavior of such materials requires observations from many directions and knowledge about the light condition. While it is possible to measure or estimate the material parameters of single objects, we are far from adequately capturing the parameters for all surfaces in the environment of a typically AR scenario.

The problems statements, discussed in Section 1.2, have been addressed by various methods over the last decades. However, none of them is solved yet completely and because of the complexity of the problems and constraints on the hardware, they will probably stay unsolved for the foreseeable future – especially when considering mobile devices and fully dynamic environments.

4.5 Rendering on Mobile Hardware

One goal of this thesis is to provide solutions for photorealistic rendering on smartphones and tablets. Therefore, it is necessary to discuss the special requirements and capabilities of these platforms. Unfortunately, there is no such thing as "the mobile platform". On one hand, the development of supported hardware features and computational power is incredibly fast. On the other hand, there is an enormous number of different devices and, as a developer for mobile applications, you need to support several hardware generations in order to compete on the market. This emphasizes the importance of scalable approaches, which can be applied platform-independently.

The developments of the last years showed, that mobile devices support more and more features, that have been exclusive to high-end desktop GPUs not too long ago. The current version of the *Standard for Embedded Accelerated 3D Graphics*, *OpenGL ES 3.2*[22], supports even *Geometry* and *Tessellation Shaders* as well as capabilities for General Purpose Computation on the Graphics Processing Unit (GPGPU) in form of *Compute Shaders*. Missing features are therefore no valid argument against advanced rendering techniques on mobile platforms anymore.

Nevertheless, there are restrictions to keep in mind while designing algorithms for such devices. The most common performance bottlenecks, as stated in the Nvidia *GameWorks Dokumentation*[23], are the following:

- The memory bandwidth is very low for devices with high-resolution screens when compared to desktop hardware. Accessing large textures or using formats with high bit-depth are expensive, especially without mipmaps or when applying higher-order filters.
- Related to the resolution, the fragment fill rate for applications using complex shaders is another bottleneck, which gets even more critical when the pixel shaders are executed multiple times for each pixel.
- Lack of CPU/GPU parallelism because of unnecessary or redundant work, done by the driver, and work, which is done on the CPU, but could be done more efficiently on the GPU.

All those issues need to be addressed on desktop hardware, too. Especially game developers are familiar with proper scene management, culling, level of detail, reducing state changes, double buffering and many more techniques,

22 The Khronos Group Inc. *Standard for Embedded Accelerated 3D Graphics*. Website, 2017.
23 Nvidia. *Optimize OpenGL ES 2.0 Performance for Tegra*. Documentation, 2007.

that are common to reduces the overall number of instruction, that are sent to driver in the first place. Because of the limited clock rates and bandwidth, but also due to energy consumption and heat production, these optimizations are even more important for the mobile platform, but eventually, they are similar to ones in desktop or console development. From a research perspective, this leads to the decision to focus on *scalable* approaches that are able to run on the recent generation of hardware, as *high-end features* of today will soon be available for the majority of devices on the market.

Detailed and most valuable guidelines for the development and performance optimization on recent hardware, can usually be found in the documentation of game engines and rendering frameworks, that support mobile devices[24].

24 Unity Technologies. *Mobile Developer Checklist*, Documentation, 2017.
Nvidia. *GameWorks Documentation*, Documentation, 2017.
Epic Games. *Unreal Engine 4: Mobile Game Development*, Documentation, 2017.

5

DISTRIBUTED NEAR-FIELD ILLUMINATION

This chapter summarizes the findings and contributions of a research project, that has also been presented at a conference and in the corresponding proceedings [Roh+14] as well as in a journal article [Roh+15]. The chapter contains more detailed explanations and additionally the proposal of a grid-based indirect illumination technique to address a previous limitation and thereby an extension of the method.

Photorealistic augmentation on mobile devices, like smartphones and tablet PCs, is not yet feasible since the computational power of the devices is insufficient for computing global illumination simulations on their own. Assuming this statement is true, a reasonable strategy is to consider multiple devices that share the effort or outsourcing the computations completely. Streaming rendered images from a powerful desktop PC, as suggested by Csongei *et al.* [Cso+14], causes a latency that is too high to meet the requirements during user interactions and for multiple simultaneous views on different clients. Since we primarily aim for applications like the augmentation of real prototypes, for which a correct illumination at any place in the scene is required, we also need continuous updates on the dynamic light condition. For a single device however, even when the limitations in computing power and the quality of sensor data might become less important factors in the future, there is always a lack of information. Not all areas of the scene are visible at once, because of a limited field of view or occlusions between different objects (see Figure 5.1). To overcome both issues, we suggest a novel, distributed approach, in which the computational effort is shared in a way that allows for dynamic light updates from all important areas of the scene and interactive low-latency rendering on one or multiple clients.

Therefore, we split the illumination into two parts: In the first part, the existing radiance values are captured by a number of HDR video cameras, that are placed at different locations in the scene, such that each part of the scene is visible to at least one camera. This acquisition process and the extraction of parameters for our lighting model is executed on a stationary PC. Based on the extracted information, we display augmentations with coherent illumination at an interactive frame rate on the mobile device, as shown in Figure 1.1. This includes consistent illumination of virtual objects with direct light, indirect light (color bleeding) and shadows of primary and strong secondary lights.

To avoid a potential bottleneck of the bandwidth between PC and mobile devices, our illumination model reduces the amount of transferred data, required for the reconstruction of the environmental lighting condition and the illumination of virtual objects.

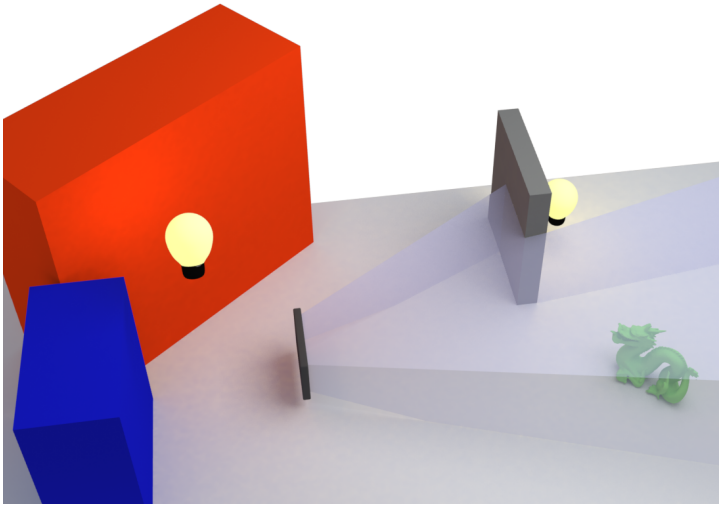


Figure 5.1: Limited Field of View and Occluders
The camera image of a mobile device does not see the important light sources required for a consistent illumination of a virtual object.

5.1 OVERVIEW

Our goal is the consistent illumination of virtual objects on mobile devices in a real environment. Multiple users should be able to interact in the real world with photorealistic augmentations. We thereby focus on an indoor scenario with a difficult, spatially and temporally varying near-field illumination (Figure 5.1). This theoretically requires the knowledge of the plenoptic function (see Section 4.1.1), which describes the real radiance values at any point in the scene, viewed from any direction at any time. Since capturing the function in its complete extent is not feasible, we approximate it by means of view-independent surface light, i.e., by dynamic texture maps. Based on this information, an interactive global illumination simulation can be computed.

5.1.1 *Hardware Setup and Precomputations*

Our hardware setup is shown in Figure 5.2: Multiple HDR video cameras are connected to a stationary PC. The cameras are equipped with fish-eye lenses and placed in the scene, such that all regions are visible to at least one camera. To enable the measurement of radiance values on real environment surfaces, each camera has to be calibrated in a pre-process. Intrinsic parameters are estimated by the method of DAVIDE SCARAMUZZA and stored in a lookup table as described in Section 3.2. To reconstruct the extrinsic parameters, we simply capture a tracked checker board and reconstruct the position of the camera. To acquire absolute real radiance values instead of arbitrary pixel colors, a photometric calibration is necessary. Therefore, we reconstruct the camera response curve using PFSTOOLS[25], leading to linear relative radiance values after applying the inverted response curve. By capturing images of an XRITE COLORCHECKER[26] and a least-squares approximation, we obtain a matrix that maps the measured colors onto reference colors, known for each

25 Max-Planck-Institut Informatik. *PFScalibration*. Project Website, 2015.
26 X-Rite, Inc. *ColorChecker Classic*. Product Website, 2017.

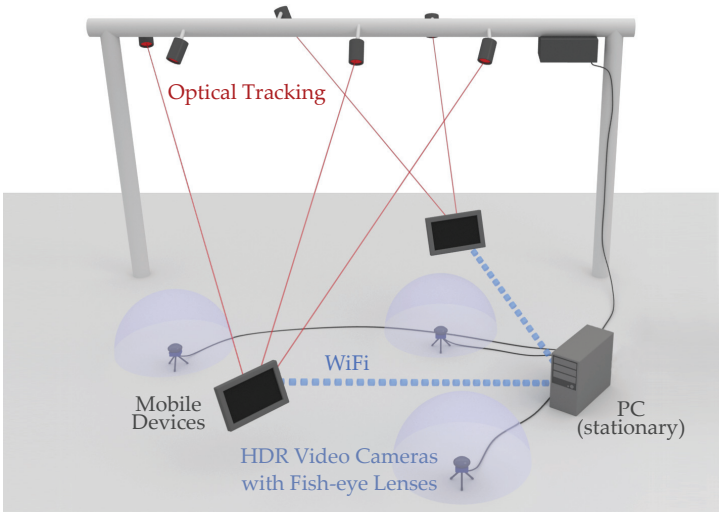


Figure 5.2: Hardware Setup

tile of the checker. A similar process is repeated for each mobile device. Additionally, we estimate another matrix to compensate the color shift introduced by the display.

The current position and orientation of the mobile device are captured at run-time. For communication between the stationary PC, mobile devices and the tracking system we use WiFi.

In a pre-process, the geometry and the diffuse materials of the real environment are reconstructed manually, using a common DCC-tool. For estimating the diffuse reflectance coefficients, we captured images of the color checker, together with the surface to measure. By assuming the irradiance is constant for the acquired area, the parameters of the target can be found as linear combination of the known reflectance parameters of the color checker. The resulting model is a very coarse representation with a simple uv-mapping that is later used as a texture atlas (Section 5.1.3).

5.1.2 *Distributed Illumination*

Given the HDR information – real radiance at each position of the environment – in combination with the 3D model, we aim for a consistent illumination of virtual objects. Based on measured radiance values of the real environment, there are different choices for interactive global illumination. The obvious solution is to use one of the methods presented in [Rit+12] to render the images on a stationary machine and stream the results to all mobile devices. We do not follow this idea for several reasons: First, we need a different image for each mobile device which can lead to a performance break-down on the server side in case of many mobile devices. Additionally, a major concern for direct user interaction is the response time of the system. This is because there is a latency in sending notifications of user input to the stationary PC, as well as waiting for the generation, compression, and transmission of the rendered image that is eventually combined with the camera image. We therefore developed an illumination model that distributes the computation between the static PC and the mobile devices.

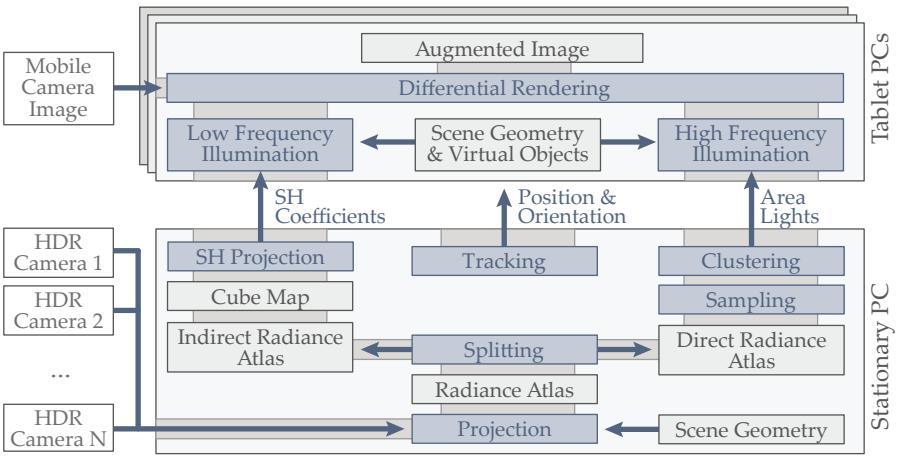


Figure 5.3: Pipeline Overview
The whole pipeline for distributed illumination from capturing multiple images of the real environment to the augmentation of the live camera stream on mobile devices.

One option for interactive global illumination is the extraction of a set of VPLs [Kel97] (see Instant Radiosity in Section 2.7.2). This allows for a complete illumination from all directions with a shadow cast by each VPL. For good quality, at least a few hundred VPLs are required. Unfortunately, only a few VPLs can be computed on a mobile device at interactive frame rates. On the other hand, PRT [SKS02] techniques can be used, which allow real-time illumination with natural light (see Section 2.7.6). These techniques work well for low-frequency illumination and diffuse materials. This is especially useful for indirect illumination, such as color bleeding from real to virtual objects. However, high-frequency illumination and hard shadows are difficult to achieve. To solve this problem, we developed a hybrid solution that combines the best of both approaches. Our solution is based on the observation that most typical settings consist of a few bright light sources and large low-frequency indirect light regions. We follow the idea introduced in [Gib+03] and split the incoming light into a high-frequency and a low-frequency part. There are two reasons for this: First, it allows for an efficient illumination with the desired effects: The high-frequency illumination and shadows can be displayed with a small set of VPLs, while the low-frequency illumination such as color bleeding can be implemented with PRT. The second reason is that this combination requires only a small amount of data that needs to be transferred between the stationary PC and the mobile devices, enabling interactive update rates. In contrast to that, Gibson *et al.* [Gib+03] create a subdivision of the environment geometry and treat the resulting patches as source and/or receiver. The lighting in the scene changes, because of occlusions – between source and receiver patches – that are caused by the insertion of virtual objects. This influence on the background is computed in the sense of differential rendering. Since we do not work with patches and links between them, we only make use of the fundamental idea to classify the regions with the highest intensity as primary light source.

(a) Synthetic Scene with Cameras (b) Real Cameras

Figure 5.4: Cameras for Scene Acquisition
Acquiring the radiance of a simple synthetic scene with three cameras (a) and the corresponding real-world camera. We are using MATRIX VISION MVBLUEFOX-IGC200 HDR video cameras equipped with 180° fish-eye lenses (b).

5.1.3 *Pipeline Overview*

The whole pipeline, from capturing images of the real world to displaying the augmented image using the distributed illumination, is summarized in Figure 5.3: The HDR video cameras with fish-eye lenses capture the existing radiance values. Then, on the stationary PC each image is projected onto the reconstructed 3D geometry using a hemispherical projection and shadow mapping. The recorded radiance values are stored in a radiance atlas, which describes a 1:1 mapping of 3D scene points to atlas texels (Section 5.2.1). To capture the illumination at all relevant parts of the environment, we use multiple cameras. Therefore, areas seen by more than one camera, receive multiple measurements leading to a more robust result. For an illumination at both, interactive speed and high quality on a mobile device, we proceed as follows: The radiance atlas is split into two parts: A direct (high-frequency) radiance atlas and an indirect (low-frequency) radiance atlas (Section 5.2.3). The direct radiance atlas is transformed into a small set of area lights (Section 5.2.3), which is transferred to the mobile device. PRT is used for the remaining low-frequency illumination on the clients. Thus, the indirect radiance is projected into the SH basis (Section 5.2.4) and the resulting coefficients are transferred to the tablet PC as well. Based on this information, the illumination of virtual objects can be computed quickly on the tablet PC – without streaming any images. Using differential rendering, the virtual object can then be inserted into the tablet camera image with correct appearance and shadows (Section 5.3).

5.2 SERVER COMPUTATIONS

We explain the server computations of our method based on a simple synthetic example shown in Figure 5.4. The scene consists of two light sources and contains geometry that casts shadows. Hence, we have a small set of non-trivial features for testing, before evaluating the approach in a real-world scenario.

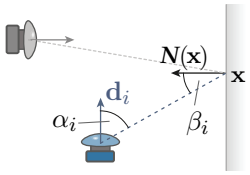


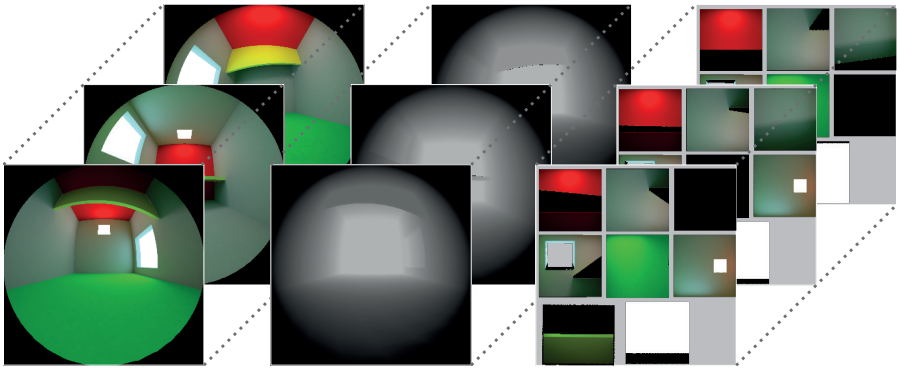
Figure 5.5: Angular Weighting
The angular weights used for merging the projected camera images.

5.2.1 *Acquiring the Radiance Atlas*

We use a texture atlas to record radiance values for all points in the scene. To update the current lighting condition, each HDR camera permanently projects the captured radiance into the atlas. This is implemented by rendering the reconstructed scene with a vertex shader that replaces the vertex position with its texture coordinates and outputs the world position along with the vertex normal to the pixel shader stage. There, we project the world position of each fragment into the camera image space to get the corresponding image coordinates. Subsequently, we sample the camera image and a previously generated artificial depth image at this location to classify the visibility of the currently processed texel in a way similar to shadow mapping (see Figure 5.6). Since triangles, that are not facing the camera, cannot be seen, they are rejected during the rendering into the atlas, depending on the dot product of the surface normal and view direction.

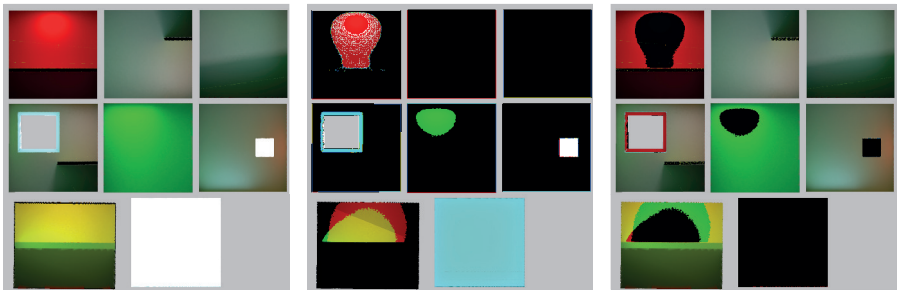
When multiple cameras see the same region, we compute a weighted average of the camera images (see Figure 5.5). To account for the low resolution in the border regions of a fish-eye projection, we use the angle α_i to the main camera direction $\mathbf{d}_i$ as a weighting. We also use the angle β_i between the view direction and the normal $\boldsymbol{N}(\mathbf{x})$ at surface position $\mathbf{x}$ to compensate inaccuracies during the reconstruction by considering steep angles less reliable. As final weight for a texel of camera i we are using:

$$w_i = \overline{\cos}\,\alpha_i \,\overline{\cos}\,\beta_i.$$



(a) Camera Images (b) Virtual Depth Renderings (c) Projection into the Atlas

Figure 5.6: Acquiring the Radiance Atlas
Each HDR camera records a fish-eye image of the scene (a). Additionally, depth buffers (b) are rendered using the reconstructed scene. The camera images are then projected into the radiance atlas (c), using the depth buffers for visibility tests. Note, that each camera contributes only partial information of the total scene radiance.



(a) Radiance Atlas (b) Direct Radiance (c) Indirect Radiance

Figure 5.7: Splitting the Radiance Atlas
The radiance atlas (a) is split in direct radiance (here stored as intensity) (b) and indirect radiance (c). Note, that the direct radiance atlas contains both, the light sources and bright indirect regions. The separation is computed per color channel to allow sources in monochrome regions, that would have a low gray-scale brightness.

The cosine is clamped to the interval $[0, 1]$ to avoid special treatment of negative values. Since each texel in the atlas can become an indirect light source, we store both, position and normal, to correctly place and rotate the light. For photometric correctness, each texel additionally stores the radiance value and the spatially varying world-space area of the texel. To compensate artifacts at texture seams, we apply a dilation over the eight neighbors with a range of two texels.

5.2.2 *Splitting the Radiance Atlas*

To determine the radiance L at any point $\mathbf{x}$, seen from direction $\boldsymbol{\omega}_o$, we integrate over the upward hemisphere to solve Kajiya's rendering equation (2.12) as discussed in Section 2.6:

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = L_e(\mathbf{x}, \boldsymbol{\omega}_o) + \int_{\mathcal{H}_+} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, L(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i \; \partial\boldsymbol{\omega}_i .$$

where $L(\mathbf{x}, \boldsymbol{\omega}_i)$ is the incoming radiance at $\mathbf{x}$ from direction $\boldsymbol{\omega}_i$, f_r is the BRDF, and θ_i is the angle between $\boldsymbol{\omega}_i$ and the surface normal $\boldsymbol{N}(\mathbf{x})$. At this point we ignore the self-emission $L_e(\mathbf{x}, \boldsymbol{\omega}_o)$ and split the reflected radiance into direct radiance $L_{\mathcal{D}}$ and indirect radiance L_I:

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o) + L_I(\mathbf{x}, \boldsymbol{\omega}_o) ,$$

where $L_{\mathcal{D}}$ corresponds to the direct radiance caused by the primary light sources and strongly reflected indirect light, whereas L_I is the remaining indirect radiance. This means that we decide for each direction $\boldsymbol{\omega}_i$ whether it corresponds to incident direct or indirect radiance. In the first case, we refer to the set of those directions as $\mathcal{H}_{\mathcal{D}+}$. In the other case, we use $\mathcal{H}_{I+}$, respectively. Eventually, we can write the integral forms as follows:

$$L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\mathcal{H}_{\mathcal{D}+}} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, L(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i \; \partial\boldsymbol{\omega}_i \tag{5.1}$$

$$L_I(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\mathcal{H}_{I+}} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, L(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i \; \partial\boldsymbol{\omega}_i . \tag{5.2}$$

To implement this separation, we split the radiance atlas into a direct and an indirect radiance atlas. For this, we determine a threshold value: Texels in the atlas with a radiance larger than the threshold are assigned to the direct radiance atlas, the other texels are assigned to the indirect radiance atlas. To allow for varying lighting conditions, this threshold is adjusted dynamically. For that purpose, we provide a user-defined parameter τ that describes how much of the total amount of light in the scene should be assigned to the direct light. To determine the threshold radiance based on τ, we first compute the histogram of all radiant intensity values in the atlas. We then accumulate the radiant intensity values from high to low until the given percentage is reached. In this way, the direct radiance atlas always contains a certain amount of direct or strong indirect light. Figure 5.7 shows how this separation looks like in the synthetic example. The selection of a suitable value for τ is discussed in Section 5.5.

5.2.3 *Finding Direct Light Sources*

After splitting the atlas, we handle both parts separately, starting with the direct light that is currently stored in a discretized texture atlas. Therefore, we reformulate Equation (5.1) and express the direct reflected radiance $L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o)$, at an arbitrary surface point $\mathbf{x}$ within the scene, using a discretized form, too. Since we assume diffuse real-world materials, we can interpret each of the N texels t in the atlas as small Lambertian emitters and accumulate their contribution:

$$L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o) \approx \sum_{t=1}^{N} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, V(\mathbf{x}, \boldsymbol{\omega}_i)\, L(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i\, \Delta\boldsymbol{\omega}_i. \tag{5.3}$$

Because the emitters can be occluded from other objects in the scene, we add the binary visibility function $V(\mathbf{x}, \boldsymbol{\omega}_i)$, which takes the value 1, only if the texel t, that spans the solid angle $\boldsymbol{\omega}_i$, is visible from $\mathbf{x}$ and 0 otherwise. We now substitute the definition of $\boldsymbol{\omega}_i$, Equation (2.5), and the incident radiance by the radiance stored per texel.

$$L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o) \approx \sum_{t=1}^{N} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, V(\mathbf{x}, \boldsymbol{\omega}_i)\, L(\mathbf{x}_t) \cos\theta_i\, \frac{\Delta A(\mathbf{x}_t) \cos\theta_t}{r^2}.$$

Here, θ_t is the angle at the sender pixel and r the distance between $\mathbf{x}$ and the position of the texel, $\mathbf{x}_t$. Substituting the approximation $L(\mathbf{x}_t) \approx \frac{\Delta I_\perp(\mathbf{x}_t)}{\Delta A(\mathbf{x}_t)}$ from Equation (2.18) yields:

$$L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o) \approx \sum_{t=1}^{N} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, V(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i\, \frac{\Delta I_\perp(\mathbf{x}_t) \cos\theta_t}{r^2}.$$

To simplify the computation, we now group the N texels from the direct radiance atlas into a *low* number of $j = 1..M$ clusters, where the j^{th} cluster consists of N_j texels with non-zero intensity. In accordance with [Don+09] we denote the clusters as VALs. The cluster center is used as position $\mathbf{x}_j$ of the VAL. To numerically integrate over solid angles subtended by the pixels of each cluster, we compute the intensity $I_\perp(\mathbf{x}_j)$, the area $A(\mathbf{x}_j)$ and the average normal $N(\mathbf{x}_j)$ of the clusters as follows:

$$I_{\perp}(\mathbf{x}_j) = \sum_{t=1}^{N_j} \Delta I_{\perp}(\mathbf{x}_t) = \sum_{t=1}^{N_j} L(\mathbf{x}_t)\, \Delta A(\mathbf{x}_t)\,,$$

$$A(\mathbf{x}_j) = \sum_{t=1}^{N_j} \Delta A(\mathbf{x}_t)\,,$$

$$N(\mathbf{x}_j) = \frac{1}{A(\mathbf{x}_j)} \sum_{t=1}^{N_j} \boldsymbol{N}(\mathbf{x}_t)\, \Delta A(\mathbf{x}_t)\,.$$

Hence, the direct radiance can be approximated by summing up the contribution of all M VALs:

$$L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o) \approx \sum_{j=1}^{M} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, V(\mathbf{x}, \boldsymbol{\omega}_i)\, \frac{I_{\perp}(\mathbf{x}_j) \cos\theta_j \cos\theta_i}{r^2},$$

where $\cos\theta_i$ was moved into the numerator of the fraction to emphasize the connection to Equation (2.20). As mentioned before in Section 2.5, the evaluation of the irradiance for discrete light sources, such as our VAL, leads to singularities. Since we want to be able to show near field illumination, we need to address this issue. Therefore, we borrow the idea of form factors for diffuse radiance transfer which are used in radiosity (see Section 2.6). These form factors are used to specify the portion of the total flux of an emitter, that is transmitted to a receiver:

$$\partial\Phi_{o\to i} = \Phi_o \partial F_{oi}.$$

Wallace *et al.* derived such a form factor to compute the transfer between a surface element and a disk shaped emitter [WEH89]. Since this corresponds to the expected shape generated by the clustering and agrees with the assumption of diffuse real surfaces, we calculate the irradiance, contributed by a VAL, using their form factor:

$$\partial F_{oi} = \frac{\partial A_i \cos\theta_i \cos\theta_o}{\pi r^2 + A_o}.$$

This yields the following analytical irradiance to evaluate for each VAL:

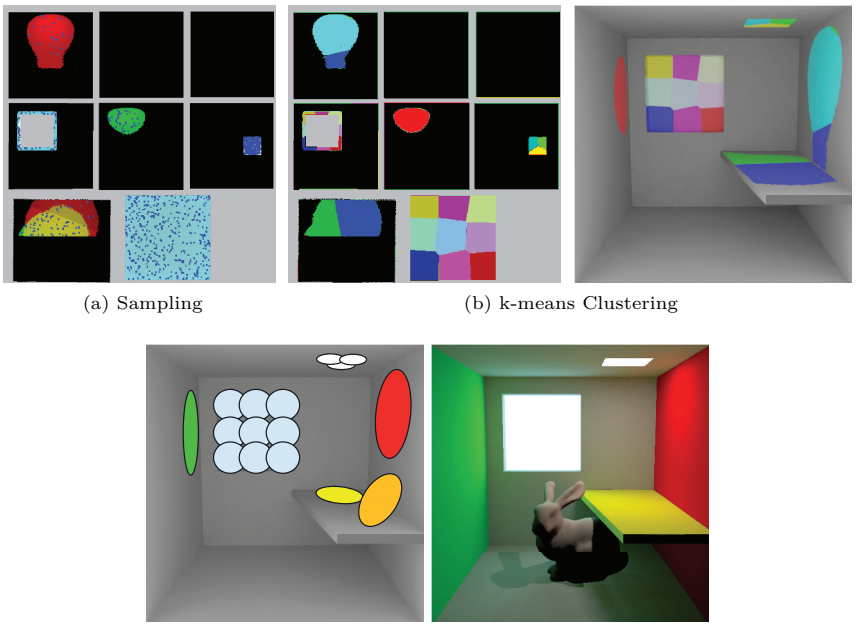
$$E(\mathbf{x}) = \frac{\partial\Phi_{o\to i}}{\partial A_i} = \frac{\Phi_o \partial F_{oi}}{\partial A_i} = \frac{\Phi_o \cos\theta_i \cos\theta_o}{\pi r^2 + A_o}.$$

After expressing the flux of the VPL as a function of the maximum intensity by using Equation (2.19), we can derive the final equation, that is used for the direct illumination:

$$L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o) \approx \sum_{j=1}^{M} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, V(\mathbf{x}, \boldsymbol{\omega}_i)\, \frac{\pi I_{\perp}(\mathbf{x}_j) \cos\theta_i \cos\theta_j}{\pi r^2 + A_j}. \tag{5.4}$$

Note, that for $M = N$, this yields Equation (5.3), where pixels are represented by small disks. For convenience, we also define the not occluded direct radiance produced by VPL j as:

$$L_{VAL_j}(\mathbf{x}, \boldsymbol{\omega}_o) = f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, \frac{\pi I_{\perp}(\mathbf{x}_j) \cos\theta_i \cos\theta_j}{\pi r^2 + A_j}. \tag{5.5}$$



(a) Sampling (b) k-means Clustering

(c) Virtual Area Lights (VALs) (d) Direct Illumination

Figure 5.8: VAL Extraction for Direct Light

Importance Sampling on the direct radiance atlas (a). Using k-means clustering, these samples are grouped in M clusters (a). Each texel is assigned to the closest cluster center. Integration over each cluster leads to M VALs (c), that are used for direct illumination of a virtual object (d).

To avoid flickering, these extracted virtual area lights have to be coherent under temporally varying illumination. To accomplish this, we modified the clustering method described by Dong *et al.* [Don+09]. Instead of generating VPLs by sampling a reflective shadow map using a Halton sequence, we draw a set of samples in the direct radiance atlas, using importance sampling on the GPU. By dividing the radiance of each pixel by the overall radiance of the direct radiance atlas we can define a PDF. Prefix sum scans are then used to generate a CDF, which is sampled by applying a 2D variant of the inverse CDF method (also see Section 2.6.2).

Similar to Dong *et al.*, we use k-means clustering on the sample positions and apply weights based on the normal at each sample. To properly compute the radiant intensity $I_{\perp}(\mathbf{x}_j)$, area and surface normal per VAL, each texel in the direct radiance atlas is assigned to its closest cluster center, using the same distance metric. Figure 5.8 visualizes the sampling and clustering results and shows the VPLs as well as the direct illumination computed by evaluating them.

At this point, we skipped the visibility computation, which will be explained in Section 5.3. There, the visibility of a VAL is approximated by a single pair of shadow map lookups, which also follows Dong *et al.* [Don+09]. To get a correct visibility estimation, all surface locations in the clusters have to be evaluated instead.

The data to be transferred to the mobile device for each VAL is the following: Position (12 bytes), normal (4 bytes, compressed), radiant intensity (12 bytes) and area (4 bytes). In total, these are only 32 bytes per VAL. The total number of VALs is a time-quality trade-off. In our experiments we use $8 \leq M \leq 64$.

Note, that we only use the direct radiance atlas for the VAL extraction and ignore the current position of the virtual objects. As an alternative, an environment map could be rendered from the virtual object center. The drawback of this option is that we might miss some important light sources, which are not visible from the center of the virtual object.

5.2.4 *Compressing Indirect Light*

For the indirect light $L_I(\mathbf{x}, \boldsymbol{\omega}_o)$, we assume that the remaining illumination in the indirect radiance atlas is of low frequency. In this case, a compression, using spherical harmonics, can be applied to the environment light around the virtual object. For a diffuse virtual object with reflection coefficient ρ_d, it is sufficient to use only the first three bands and thereby $K = 9$ basis functions for an not occluded illumination with a barely visible error, as shown in [RH01b, SKS02]. Please see Section 2.7.6 for more details. Given a vertex v at position $\mathbf{x}_v$, the indirect radiance is computed by a simple dot product:

$$L_I(\mathbf{x}, \boldsymbol{\omega}_o) \approx L(\mathbf{x}_v, \boldsymbol{\omega}_o) \approx \frac{\rho_d}{\pi} \sum_{i=0}^{K-1} c_{li}\ c_{vi}, \tag{5.6}$$

where the coefficients c_{li} and c_{vi} are obtained by a projection onto the SH basis function y_i:

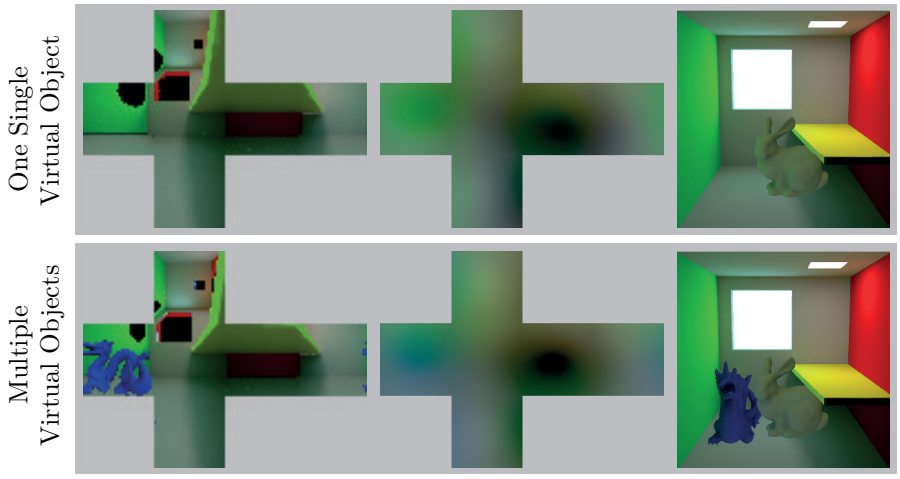
$$c_{li} = \int_{\mathcal{H}_{I_\pm}} L(\mathbf{x}_c, \boldsymbol{\omega})\, y_i(\boldsymbol{\omega})\ \partial\boldsymbol{\omega} \tag{5.7}$$

$$c_{vi} = \int_{\mathcal{H}_\pm} V(\mathbf{x}_v, \boldsymbol{\omega})\, \overline{\cos}\,\theta\ y_i(\boldsymbol{\omega})\ \partial\boldsymbol{\omega}. \tag{5.8}$$

The c_{li} coefficients can be interpreted as the amount of light that is incident at a position $\mathbf{x}_c$ from all directions of the surrounding. The position is referred to as $\mathbf{x}_c$, since it defines the center point of the environment map, used as input for the projection. Accordingly, the coefficients c_{vi} at a vertex v describe a set of directions from which incident light is not occluded. As the SH basis functions are orthonormal, the dot product of both is the amount of incident and not occluded light from all directions at the vertex.

The coefficients of the transfer function c_{vi} are static, so they can be precomputed and stored per vertex with the virtual object. In contrast, the coefficients c_{li} of the environment map $L(\mathbf{x}_c, \boldsymbol{\omega})$ change whenever the incoming illumination changes. We therefore render a low-resolution $(6 \times 32 \times 32)$ cube map from the virtual object center, using the indirect radiance atlas as texture of the surrounding scene. The resulting environment map is then projected to the first nine spherical harmonic basis functions y_i and the coefficients c_{li} are transferred to the mobile device. Using RGB float values, these are only $9 \times 3 \times 4 = 108$ bytes in total. In case of multiple mobile devices, the same coefficients can be re-used. In case of multiple virtual objects, this process is repeated for each object. The computation cost for this step is small (see Section 5.4).

In fact, we are able to also consider the indirect light transmission between virtual objects with a small overhead by including the other virtual objects during the indirect light estimation of the one that is updated. This is shown in the bottom row of Figure 5.9, where the shading of the BUNNY is influenced by the blue DRAGON. To keep the additional effort low, the objects are illuminated by PRT, too. Therefore, we render another cube map, containing direct and indirect light to derive SH coefficients with the correct (total)



(a) Indirect Radiance Map (b) SH Compression (c) Indirect Illumination

Figure 5.9: Indirect Light Compression
To estimate the indirect illumination of a virtual object, we first render a cube map with the indirect radiance from the object center position $\mathbf{x}_c$ (a). This is projected into the first nine SH basis functions (b), which allows a real-time illumination of a virtual object (c). The second row shows multiple virtual objects with mutual interreflections, like the blue color bleeding from Dragon to Bunny.

amount of light. In summary, we are rendering $2n$ cube maps, each containing $n-1$ virtual objects and the reconstructed scene in order to achieve additional indirect light transmission between n virtual objects. Note, that these interreflections cover $b-1$ diffuse bounces for objects that are static for b iterations (consecutive frames).

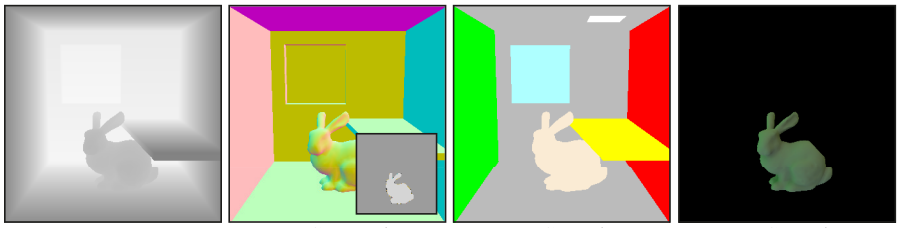
To meet the real-time requirements, we need to be able to perform this compression for multiple cube maps in a narrow time frame. Therefore, we pre-compute SH-coefficient weights w_{ti} for each cube map texel $\boldsymbol{\omega}_t$ and use the GPU to weight and accumulate the radiance $L(\mathbf{x}_c, \boldsymbol{\omega}_t)$ per texel:

$$\begin{aligned} w_{ti} &= y_i(\boldsymbol{\omega}_t)\ \boldsymbol{\omega}_t \\ c_{li} &\approx \sum_{t=1}^{6\times32\times32} L(\mathbf{x}_c, \boldsymbol{\omega}_t)\, w_{ti}. \end{aligned} \tag{5.9}$$

Therefore, the nine floating point weights per cube map texel w_{ti} are stored in a texture. Note, that Equation (5.9) represent a discretization of Equation (5.7), that can be mapped to a fast reduce program on the GPU.

The same compute program can be used to estimate c_{vi}, defined by Equation (5.8). Here, we render a cube map at each vertex v. To estimate the visibility $V(\mathbf{x}_v, \boldsymbol{\omega})$, the object is colored black and rendered into a white cube map. Additionally, we account for $\cos\theta$ by using the angle θ between the vertex normal and the direction of $\boldsymbol{\omega}$ leading to Equation (5.10). Note, that the cosine is clamped to the interval $[0, 1]$ to integrate the visible hemisphere only.

$$c_{vi} \approx \sum_{t=1}^{6\times32\times32} V(\mathbf{x}_v, \boldsymbol{\omega})\, \overline{\cos}\,\theta\ w_{ti}. \tag{5.10}$$



D32	R10G10B10A2	R8G8B8A8	R8G8B8A8
(a) Depth	(b) Normal + Flags	(c) Reflectance	(d) Indirect Radiance

Figure 5.10: G-Buffer
The off-screen buffer, containing geometry and material information of the reconstructed as well as the virtual scene per pixel.

5.3 RENDERING ON THE CLIENT

Due to the described separation of illumination, the final image generation on the client only requires lightweight operations on the mobile device with limited rendering capabilities: For each VAL, we compute the direct radiance and visibility, using shadow mapping as specified in Equation (5.4). We then add the indirect illumination, which is computed per vertex by applying Equation (5.6), using the stored coefficients c_{vi} of the model and the transferred coefficients c_{li} for the indirect illumination. To display virtual shadows with correct brightness, we use differential rendering (see Section 4.2) and subtract the direct radiance in regions of virtual shadows.

To improve the rendering performance, we use a *tile-based deferred shading* based on ANDERSSON [And09] (see Section 2.7.1). Compared to simple forward rendering and non-tiled deferred rendering, a tile-based approach reduces overdraws to a minimum, because each final screen texel is processed only once. Additionally, the G-Buffer (see Figure 5.10) needs to be read only once, which improves performance, since memory accesses are expensive.

In the first pass, the reconstructed and the virtual scene are rendered into one G-Buffer, containing projection space depth, world space normal, diffuse reflectance coefficients, and indirect radiance as well as flags to distinguish virtual from real objects (see Figure 5.10). To avoid unnecessary geometry processing, we calculate the indirect radiance for virtual objects by PRT in the vertex shader. Hence, we need to render the scene only once, except for the shadow map generation described later in this section.

The second pass handles light calculations and the composition of the augmented image in one single compute shader program. Thus, the screen is divided into tiles of 8×8 texels – which performed best in our tests. Each tile is processed by a thread group of 64 threads and each thread executes the following steps:

1. Read the background image and G-Buffer data for the corresponding texel and construct the view frustum around the tile. Near and far plane are determined by the minimum and maximum occurring depth values within the tile.
2. Cull the VAL assigned to the thread, if the view frustum is entirely in the negative hemisphere of the VAL. Due to the Lambertian emission of the VALs, such a VAL does not contribute to the illumination of the tile. Because of the group size, 64 VALs can be treated simultaneously. If there are more VALs than threads per tile, this process is performed

in a loop. Lights not culled are added to a shared list, eventually containing $m \leq M$ visible VALs.

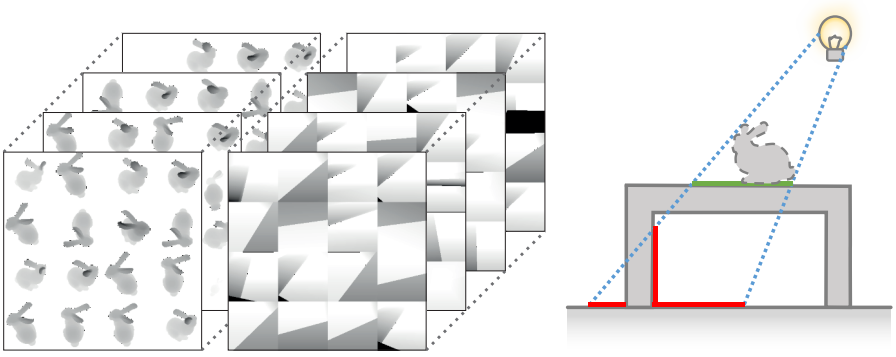
3. Perform visibility and shading operations to illuminate the surface position $\mathbf{x}$ at the texel by all remaining m VALs in the group shared VAL list. For differential rendering, we accumulate the radiance L_{VAL_j} of all VALs depending on the texels' flags. For texels marked virtual, we store radiance that is not shadowed, neither by real nor by virtual objects. For non-virtual texels, we store radiance that is shadowed by virtual but not by real objects. In essence, we estimate the light that should be missing because of new virtual shadows.
4. Combine the results of step 3, the background color L_b and the indirect radiance $L_I(\mathbf{x}, \boldsymbol{\omega}_o)$, using Equation (5.11) for texels marked and not marked as virtual. The visibility at $\mathbf{x}$ from VAL j in the reconstructed scene is referred to as $V_j(\mathbf{x}_v, \boldsymbol{\omega})$, whereas $\hat{V}_j(\mathbf{x}_v, \boldsymbol{\omega})$ is the visibility in the virtual scene.

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = \begin{cases} L_I(\mathbf{x}, \boldsymbol{\omega}_o) + \sum_{j=1}^{m} V_j(\mathbf{x}, \boldsymbol{\omega}) \;\; \hat{V}_j(\mathbf{x}, \boldsymbol{\omega}) \;\; L_{VAL_j}(\mathbf{x}, \boldsymbol{\omega}_o) & \text{if virtual} \\ L_b - \sum_{j=1}^{m} V_j(\mathbf{x}, \boldsymbol{\omega}) \;\; (1 - \hat{V}_j(\mathbf{x}, \boldsymbol{\omega})) \;\; L_{VAL_j}(\mathbf{x}, \boldsymbol{\omega}_o) & \text{otherwise.} \end{cases} \tag{5.11}$$

We use two shadow maps per VAL to cover shadows from reconstructed and virtual objects [Kne+12, GM00]. This is necessary to prevent virtual objects from casting shadows through real objects (see Figure 5.11b). Using only one shadow map, containing the closest distance in light space, can lead to correct shadows (green). But without the distance of the closest reconstructed object we are not able to identify the correct shadow receiver and add wrong shadows (red) on every further real surface. Hence, we need $2M$ shadow maps for direct illumination with M VALs, which is not feasible for large M in real-time. To reduce the geometry processing overhead we update 16 shadow maps at once by using a geometry shader for duplicating the primitives and for rendering to multiple viewports simultaneously. Therefore, we organize our shadow buffer in a texture array containing 4×4 shadow maps per slice (see Figure 5.11a). To adjust to the narrow time budget, we update only one slice of the virtual shadow buffer and one slice of the reconstructed shadows per frame. The update order follows *Round-robin*, but recently updated VALs are preferred. To obtain good shadows with low resolution, we construct each shadow frustum to closely fit all visible virtual objects and use this frustum for both virtual and reconstructed shadow maps. To increase depth precision, the near plane is set to be close to virtual object and the far plane is limited to just contain the reconstructed scene. The reconstructed geometry between light and near plane is projected onto the near plane.

5.4 RESULTS

In this section, we report the results that are obtained by our distributed approach for augmenting live camera streams with virtual objects, illuminated by the dynamically captured real-world environment. All performance experiments for rendering were run on a MICROSOFT SURFACE PRO with INTEL I5-3317U CPU, 1.7 GHz, 4 GB RAM and INTEL HD GRAPHICS 4000. The stationary PC used for image acquisition and calculation of the light model



(a) Shadow Map Buffers (b) Double Shadowing

Figure 5.11: Visiblity Estimation by Shadow Mapping
Two shadow maps per VAL (a) are required to avoid double shadowing (red) from virtual objects (dashed contour) in regions that are shadowed in the real environment (b).

parameters was equipped with an AMD PHENOM II X4 965 CPU, 3.4 GHz, 8 GB RAM and a NVIDIA GEFORCE 580 GTX.

For comparison in quality and performance we decided to evaluate the synthetic CORNELL SCENE used in Section 5.2 to avoid inaccuracies caused by the camera sensors and lenses as well as the reconstruction process. Nevertheless, results of real-world scenarios are demonstrated later in this section. The scene was designed to be as simple as possible, while showing the most important interactions between real and virtual objects. In particular, there are shadows from virtual on real objects and vice versa, virtual objects occluding real objects and vice versa, and there is a strong indirect light that causes color bleeding. The radiance of the small light at the ceiling is $15\,\mathrm{W/sr\,m^2}$ and thereby five times brighter than the window with $3\,\mathrm{W/sr\,m^2}$. The scene is augmented by a BUNNY with 2.5k triangles and the resolution of the G-Buffer is 960×540, if not otherwise stated.

5.4.1 *Comparison*

To evaluate our approach we compare it with a standard VPL-based lighting, PRT and different combinations of light clustering and splitting into direct and indirect light. To achieve fair results, we use the same renderer with all optimizations by tiled rendering and simplifications during shadow map updates for rendering VPLs as we do in our case. For PRT we were also using the G-Buffer to treat occlusions between real and virtual objects as well as the same calculation of indirect light used in our approach, but we disabled all direct light calculations and shadow map updates since they cannot be used with PRT.

Figure 5.12 shows results of the different methods depending on the number of direct light sources. For the synthetic scene we created a path traced reference image, depicted in the lower right corner. Next to this ground truth solution, the result of simple PRT without any directional lights is shown.

In the first column, the classical Monte Carlo-based VPL lighting, as described in Section 2.7.2, is depicted. To generate VPLs, the radiance atlas was sampled, using the gray scale intensity as density function p. While the

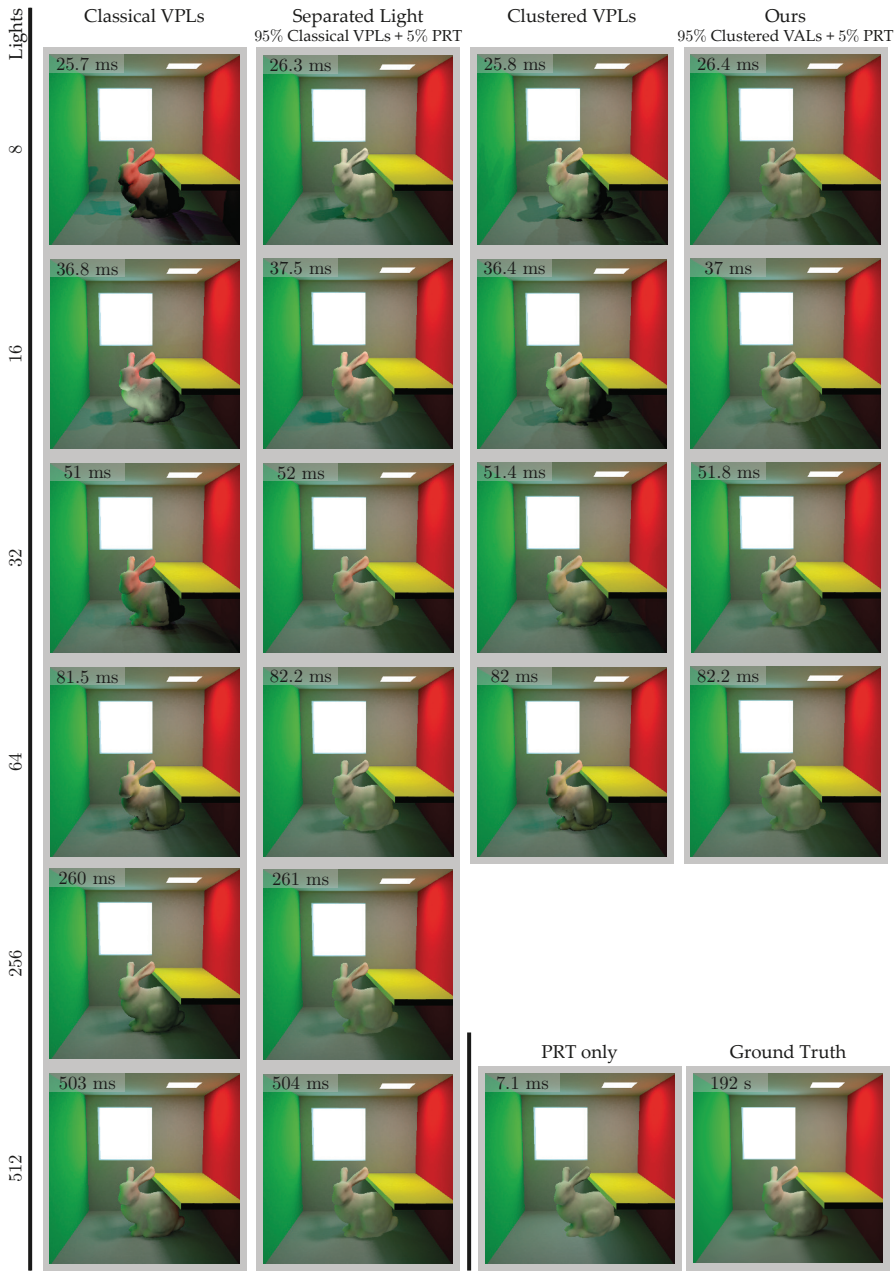


Figure 5.12: Results of the Synthetic Test Scene

A comparison of the different techniques and combinations in terms of visual quality and performance, depending on the number of lights. The techniques are also compared to a path-traced ground truth in the bottom right corner and to a purely PRT-based solution without shadows.

position is directly read from the atlas at sample position $\mathbf{x}_s$, the maximum intensity of the resulting VPL is calculated by:

$$I_{\perp VPL}(\mathbf{x}_s) = \frac{1}{M_{VPL}} \frac{I_{\perp}(\mathbf{x}_s)}{p(\mathbf{x}_s)},$$

where M_{VPL} is the number of VPLs (and samples), $I_{\perp}(\mathbf{x}_s)$ is the intensity stored in the atlas and $p(\mathbf{x}_s)$ is the probability to produce the sample at position $\mathbf{x}_s$. The VPLs are assumed to be small Lambertian emitters too, but without any spatial extent.

The second column combines the classical VPL lighting with our light separation. Instead of sampling the complete radiance atlas, we just sample the direct light atlas (see Figure 5.8a). The indirect light is compressed to spherical harmonics, as we do in our approach.

The third column shows clustered VPLs without light separation. In this case, we apply our direct light clustering step to the classical VPL method. Here we draw 4k samples, cluster them by k-means and integrate the radiance atlas, which results in one VPL for each cluster, similar to the description in Section 5.2.3, but without the disc form factor.

The last column contains the final results of our approach with light separation and clustered VALs.

5.4.2 *Evaluation of the Visual Quality*

In comparison with the ground truth image, the result of the PRT method shows significant differences. Besides the lack of shadows, there is a visible shift in the color of shading. The environment coefficients used for this image were derived from a cube map rendered at the object center from where the bright red wall is only slightly visible. This explains the cold tone of the image and why PRT alone is not a good choice for near-field illumination, even though the measured timings are best.

Evaluating the classical VPL approach confirms the expected behavior known from instant radiosity implementations [Kel97, Kne+10, Kne+12]. A large number of lights is required to converge to the correct solution. We stopped at 512 sources, which produced a result close to the reference image in 503 ms.

By separating high-frequency from low-frequency light, the region to sample becomes smaller. In combination with PRT-based low-frequency illumination, the visual quality of the results increases, especially for a low number of light sources. Because of the smaller sample regions, the point lights concentrate in bright areas, which leads to more plausible shadows as a second benefit. The additional computation cost for PRT lighting is constant and rather low compared to the direct lighting. Note, that these two methods without light clustering are not coherent over time for smaller light counts. This results in a distracting flickering and is not suitable in most scenarios. To provide an impression, we refer to the accompanying video in Appendix A.1.

The experiments with clustered VPLs showed an improved spatial coherence but did not lead to a temporally coherent illumination because of the large cluster sizes that need to cover the whole radiance atlas. For the client, there is no difference to classical VPLs in terms of calculations for lighting, which was confirmed by equal timings.

Table 5.1: Results of the Synthetic Test Scene

Timing breakdown in ms for the stationary PC at an atlas resolution of 1024×1024, 4 HDR cameras, 4k direct light samples and 16 clusters with 20 iterations per clustering step. Updating the geometry information in the atlas is only required for dynamic scenes, $(\cdot)^*$. Steps marked by $(\cdot)^{**}$ are related to the camera images and are processed only on new incoming frames.

Computation Step	Time in ms
Update Atlas	
Position, normals and area	0.50*
Dilation	1.33*
Acquiring the Radiance Atlas	
Acquire color image	2.35**
Render depth image	0.26**
Project into atlas	0.44**
Combine radiance atlas	0.58
Splitting the Radiance Atlas	
Find separating threshold	9.10
Split into direct and indirect atlas	0.53
Finding Direct Light Sources	
Sampling (4k Samples)	6.67
k-means clustering ($M = 16$)	7.30
Integrating cluster radiances	17.50

The results of our approach, depicted in the last column, contain features of both improvements. The separation of the radiance atlas leads to smaller areas to be sampled, hence the light sources concentrate in the brightest regions. The additional clustering leads to coherent light positions and thus coherent virtual shadows. It also allows to integrate the area per light and to approximate the shape by a disc. Hence, virtual objects, close to light sources, do not show the singularities of classical point lights. Considering the measured timings, there is no difference when compared to the approach in row two, since there is no difference in rendering on the client side. Comparing the images, created with varying numbers of VALs, reveals only slight differences in the shading of the virtual object. The most obvious distinction can be found in the quality of the shadows, especially at the transition from the virtual to the real shadow cast by the yellow board. A drawback of our approach can be observed in the shadowed region on the back of the Bunny which is too bright in comparison with the reference image. One reason for that is the limited number of SH bands and the lack of details in the reconstructed indirect light. Another influencing factor is that the cube map is only valid for the center of the virtual object. Other locations on the virtual object, e.g., below the yellow board, have a slightly different environment illumination. This problem can be addressed by evaluating the indirect light at multiple locations and interpolating the SH coefficients per vertex during rendering, which leads to an approach similar to irradiance volumes [Gre+98, Gib+03]. Finally, there is another aspect that contributes to a too bright indirect lighting. The indirect radiance estimation at the center of the virtual object by cube map rasterization does not consider the shadows cast by any virtual objects.

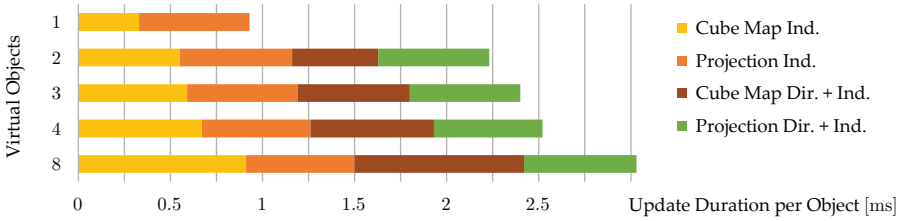


Figure 5.13: Computing SH Coefficients for Multiple Objects
Timings for updating the indirect light coefficients per object. Measured in the synthetic Cornell Scene with multiple virtual Bunnies.

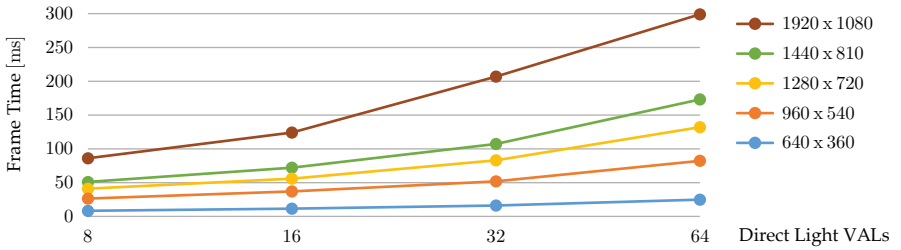


Figure 5.14: Timings with Respect to Screen Resolution
Influence of the G-Buffer resolution on the frame time. Timings measured for augmenting the synthetic Cornell Scene with the virtual Bunny.

5.4.3 *Light Extraction Performance*

All tasks executed by the server are implemented on the GPU. Since the results are transmitted asynchronously, the computations do not affect the rendering performance, discussed in the next section. However, an interactive update rate improves the visual quality of the rendering while moving virtual objects and reduces the time that is needed to respond to changes in the dynamic environment. Table 5.1 contains the timings measured with our current implementation. The individual steps are not executed in the listed order. The first block is only required if the tracking system reported a moving real object. The operations on the camera images in the second block are applied only if a camera captured a new image during the last iteration. The latter is influenced by the type and number of cameras used, their resolution, and the available bandwidth for the transfer to the GPU.

The time to update the indirect light coefficients depends on the number of virtual objects. Figure 5.13 illustrates the increasing effort with a growing number of virtual objects. As described in Section 5.2.4, we include other virtual objects to take account for indirect transmission between the objects. Hence, the time required for rendering a cube map increases with the number of objects while the duration for compressing the cube maps into SH-coefficients is constant. Note, that the time to update the indirect light without interaction between virtual objects will be equal to the duration measured for one virtual object. The rendering of an extra cube map with direct and indirect light is not necessary in this case.

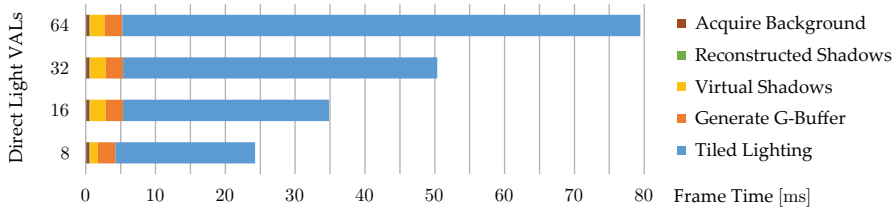


Figure 5.15: Timings with Respect to VALs Count

Timings with respect to the number of VALs measured for augmenting the synthetic Cornell Scene with the virtual Bunny, broken down and accumulated.

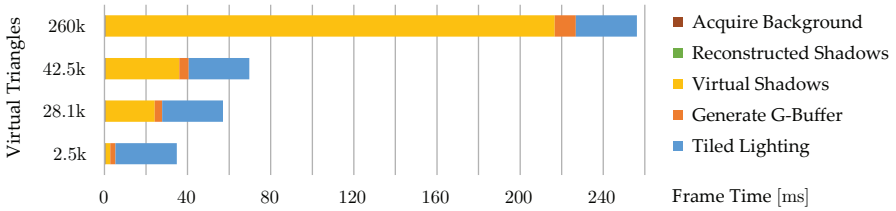


Figure 5.16: Timings with Respect to Model Size

Timings with respect to vertex count measured for augmenting the synthetic Cornell Scene with different virtual models and 16 lights broken down and accumulated.

5.4.4 *Rendering Performance*

As noted in Section 5.3, the tile-based approach reduces the number of geometry processing passes, overdraw, and G-Buffer accesses per texel. Nevertheless, the G-Buffer resolution is the most dominant factor on the rendering performance. Figure 5.14 illustrates the increasing time per frame with growing light count and resolution. In consequence of the tiled rendering the timings grow almost linearly with the number of rendered tiles.

In Figure 5.15 the frame timings are broken down to five steps. The acquisition of the background image, the rendering of reconstructed shadows, and the generation of the G-Buffer are independent of the number of VALs. The update of 16 virtual shadow maps takes 2.2 ms, if 16 or more lights are present. The largest part of the time is spent on calculating the direct illumination and visibility. After a constant offset, the required time increases linearly with the number of lights.

Because of the deferred rendering, the impact of geometry complexity on performance is assumed to be low, as each model has to be rendered only once to generate the G-Buffer. The result of the evaluation with virtual models of different complexity is depicted in Figure 5.16. As anticipated, the time required to create the G-Buffer increases with the number of primitives, up to 10 ms for $260k$ triangles. The other step that needs to render the virtual geometry is the update of the virtual shadows. Since we are processing 16 shadow maps per iteration, the duration grows faster up to 216 ms for the largest model. Fortunately, this large number is not relevant in practice, since low-poly models can be used for rendering shadow maps of low resolution. Hence, a few hundred primitives are sufficient for the 128×128 shadow maps we use in our examples, and highly detailed models are only required during the G-Buffer generation.

5.4.5 *Real-world Scenarios*

To measure the real-world radiance values, we use HDR video cameras with 180° fish-eye lenses, the MATRIX VISION mvBLUEFOX-IGC200. For tracking, we use OPTITRACK with 12 infrared cameras, capturing a range of approximately 3×2 meters.

Figure 5.17 and 5.18 contains an overview of an acquired real scene. The first row of Figure 5.17 shows the input of three different cameras. To illustrate the corresponding reconstructed scene, we rendered the wire-frame model as overlay. The second row shows the projections of the camera images into the atlas, where flaws in the reconstruction and the registration of the cameras become visible. The lower figure shows the weighted average of the project camera images (Figure 5.18a), the direct light atlas (Figure 5.18b) and the indirect light atlas (Figure 5.18c). Note, that some ghosting artifacts caused

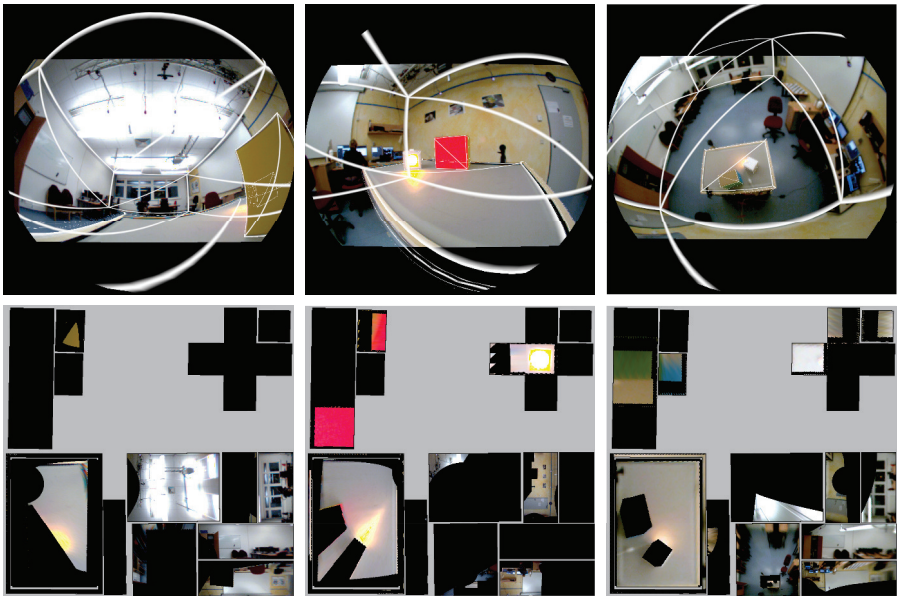
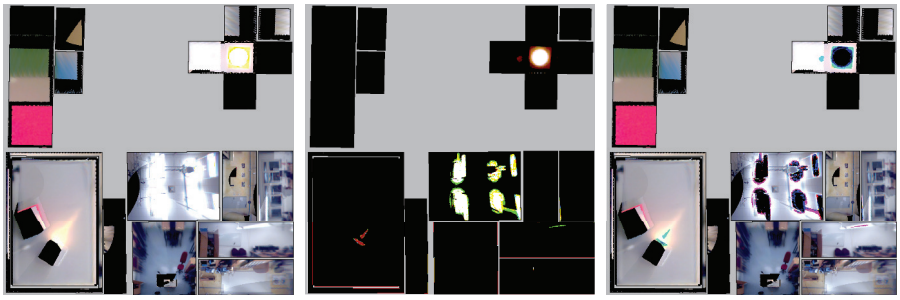


Figure 5.17: Capturing Real-world Scene Illumination
Real scene acquired by three cameras (first row). Images projected into the atlas (second row).



(a) Merged Radiance (b) Direct Radiance (c) Indirect Radiance

Figure 5.18: Splitting into Direct and Indirect Radiance
Atlas projections visible in Figure 5.17 merged into one atlas (a). Resulting direct (b) and indirect (c) radiance atlas after splitting.

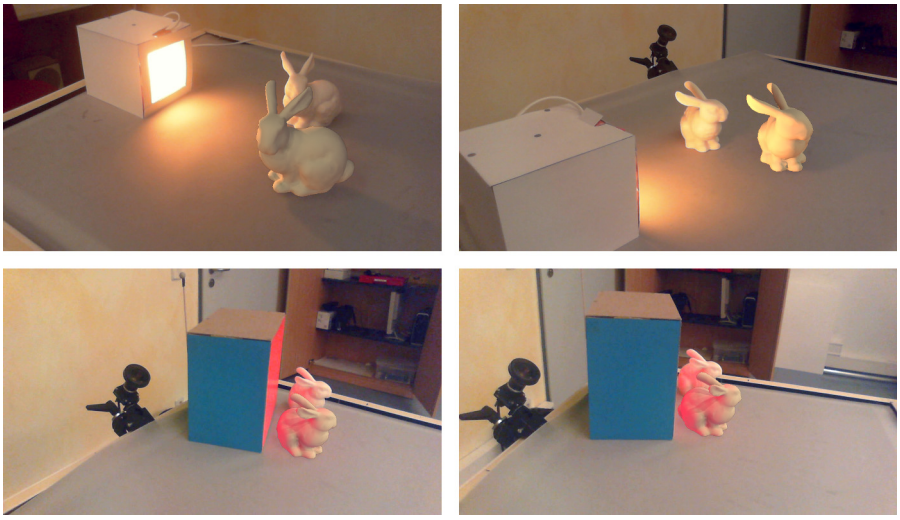


Figure 5.19: Comparison of Real and Virtual Objects
A virtual Bunny in front of its real counterpart, illuminated by a local light source (top) and a strong indirect light (bottom). In both cases, the sender becomes invisible after user movement.

by the mentioned inaccuracies become visible, but their influence on the final result is low as long as the receiver material is diffuse and the overall radiance matches the real light conditions.

Figure 1.1 in Chapter 1 shows the consistent appearance of a 3D-printed and a virtual Bunny side-by-side. We added a real and a virtual color checker to show the quality of the reproduced colors. To visually verify a correct capturing process of the *near-field* illumination, we place the Bunny close to a local light source and a strong indirect light, as shown in Figure 5.19. Images from the interactive session, visible in the accompanying video (see Appendix A.1), are shown in Figure 5.20. The sequence demonstrates both, temporally and spatially varying illumination. The performance in real and synthetic scenes has been very similar in all our experiments. This is because the cost for transferring the mobile camera image to GPU memory and the cost for rendering the synthetic background image compensate each other.

5.4.6 *Non-Diffuse BRDFs*

Because of the compression used for the indirect light, we are limited to diffuse BRDFs. However, this limitation can be ignored to enable non-diffuse materials by accepting a result that is physically not completely correct.

In Figure 5.21 different non-diffuse BRDFs are used to show that the augmentations are still plausible. The glossy Dragon on the left has a Blinn-Phong material based on the BRDF presented by Naty Hoffman [Hof+10]. The Otto bust on the right is shaded with a modified version of our Blinn-Phong material, using a rescaled Schlick-approximation of the Fresnel term [LS05] and the Bunny in the center is rendered with an image space effect to convey the impression of glass. Each of these approximations are explained in the following paragraphs.

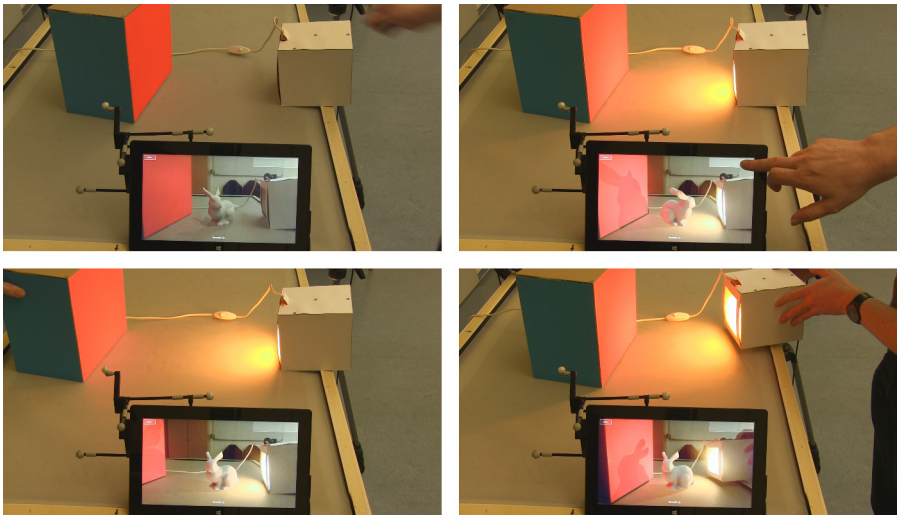
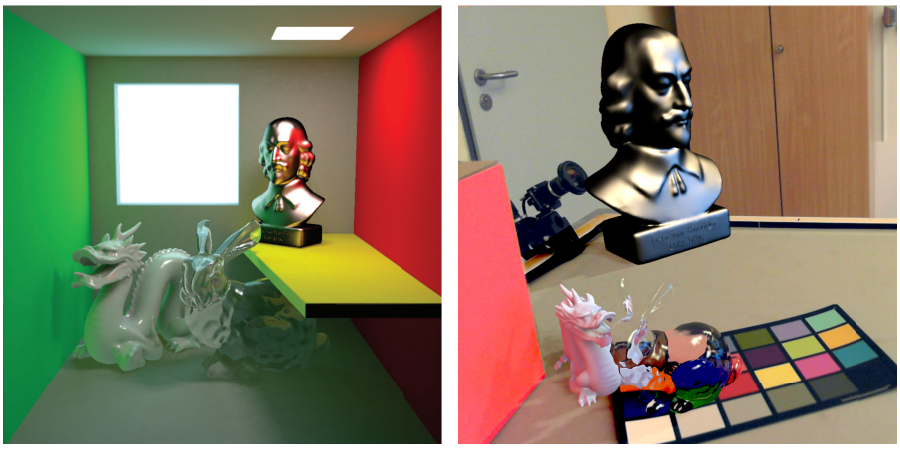


Figure 5.20: Virtual Object in Dynamic Environment
Moving tracked objects: Initial configuration (top left), user switches on the light and rotates the BUNNY at 27 Hz (top right), red color bleeding disappears, when the box is moved away (bottom left) and direct light changes after light movement (bottom right).

GLOSSY AND METALLIC MATERIALS To be able to support different shading models, a few modifications to the rendering pipeline are required and the G-Buffer needs to store additional properties. The VALs are still used for direct rendering but this time the reflectance can change per object or, more precisely, per texel of the G-Buffer. The diffuse indirect lighting, computed by PRT, is still available and can be used by the different materials if desired. In contrast to pure diffuse materials, where the irradiance from each VAL can be computed analytically and the amount of reflected light is defined by the constant reflectance coefficient ρ_d, the reflection on glossy materials is view-dependent and thereby more complex. For a correct solution one needs to integrate over the area light source and accumulate the reflected radiance, because the BRDF is not constant anymore. To meet the real-time performance requirements, we only use the center of the VAL as a representative sample. For the glossy BRDF, we use this approximation and apply the diffuse lighting by PRT, leading to a multi-layered material.

For glossy and metallic materials, we add another G-Buffer layer to store specular reflection coefficients and a roughness value. The rescaled Schlick model involves refractive indices with real and imaginary parts, instead of diffuse and specular coefficients. Thus, these slots can be used to store the index of refraction for the wavelength of each color channels. Since the indices are not necessarily in the interval $[0, 1]$, the texture formats of these two G-Buffer layers are changed to $R16G16B16A16$. During the G-Buffer generation, we make use of *Dynamic Shader Linkage*, where each material is represented by a class that fills the buffer with properties required by that material. The free alpha channel of the diffuse layer is used to store an ID to identify the material per texel. During the illumination step of the deferred lighting, the same classes are used to evaluate a BRDF function based on the stored ID. This approach allows to apply various different BRDFs, as long as the parameters fit into the G-Buffer and the objects are fully opaque.



(a) Synthetic Scene (b) Real-world Scene

Figure 5.21: Evaluating non-Diffuse BRDFs
A synthetic (a) and a real scene (b) augmented by objects with non-diffuse BRDFs. Both images show a glossy DRAGON, a glass BUNNY and a metallic OTTO bust.

Note, that due to the missing high-resolution environment images on the tablet, we cannot display highly glossy virtual objects. Instead of reflections of all parts of the real environment, only the highlights of the extracted VALs are visible. Since metallic materials reflect incoming light at the surface, there is no diffuse reflection which technically is a coarse approximation of sub-surface scattering. Hence, we do not apply diffuse illumination by PRT in this case, which is the reason for the darker backsides of the busts. As noticeable on the left of the dragon in Figure 5.21, the indirect illumination for glossy materials is still only diffuse and thereby not plausible for glossy objects.

TRANSLUCENT MATERIALS Non-opaque objects are more complex, because another layer is required to allow translucent objects in front of opaque virtual or real surfaces. For the glass shader, the G-Buffer is extended by two depth layers, one for front and one for back faces to estimate the thickness of the glass object. It is also required to introduce another normal buffer to calculate direct illumination, reflection and refraction on the glass surface. The free alpha channel is used to store a flag that indicates whether there is a translucent object at that certain texel or not. If the flag is set during the illumination step, we calculate the direct lighting for the second layer, too and write the accumulated radiance into another output buffer. Note, that the opaque objects are processed as before without any influence of this second layer.

Finally, a post-effect is applied to the augmented image, already containing opaque virtual objects. For this effect, we evaluate the reflection and refraction ray for each texel with the translucency flag set. Instead of using a more correct ray marching approach, we simply project both rays into the augmented image space and use them as offsets to sample corresponding color values. This coarse approximation results in reasonable reflections and refractions, if the sampling distance is adjusted properly and the relevant areas are visible in the image. The thickness of the translucent objects is used to estimate a transmittance T,

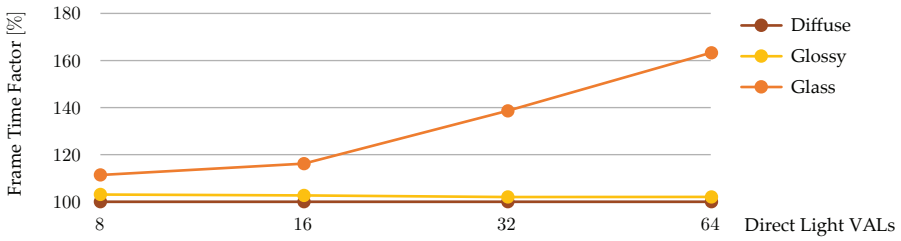


Figure 5.22: Performance Impact of non-Diffuse BRDFs
Rendering overhead on the client for augmenting the CORNELL SCENE with a glossy or a glass BUNNY compared to the former diffuse BRDF, respectively. The overhead factor was measured for different numbers of VALs for direct lighting.

based on the Beer-Lambert law. Equation (5.12) shows the final composition for marked texels, where f is the approximated Fresnel term:

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = max\big(L_{refl}(\mathbf{x}, \boldsymbol{\omega}_o)\,, L_{\mathcal{D}}(\mathbf{x}, \boldsymbol{\omega}_o)\big) + (1 - f)\; T\; L_{refr}(\mathbf{x}, \boldsymbol{\omega}_o)\,. \quad (5.12)$$

We use the maximum function to get reflections of all frequencies: from the image space approach and highlights, produced by the VALs. The latter may not be directly visible in the camera image. Note, that highlights generated by image space reflections are probably too dark, because the live camera stream is not an HDR stream and visible light sources are clamped, because of saturated texels. Additionally, the shadows of our translucent objects are incorrect, and caustics, which are the result of a correct transmission of light through glass, are also not part of the approximation and thereby a topic of further investigation. Overall, this treatment of glass objects is far away from a physically correct solution but it shows how our approach can be extended to meet the requirements of applications of higher visual complexity, that can tolerate the inaccuracies.

These changes also have an impact on the performance of the rendering client. The shader classes and the changed G-Buffer layer formats, introduced for glossy and metallic BRDFs, result in a constant overhead of about 2-3 %, compared to the diffuse material. For rendering glass objects, the additional direct lighting and the post-effect are leading to an overhead, that increases with the number of VALs (see Figure 5.22).

5.5 SELECTION OF PARAMETERS

DIRECT AND INDIRECT LIGHT In Section 5.2.2 we introduced a user parameter τ to control the threshold for splitting direct from indirect light. Figure 5.23 shows the influence of the parameter selection on the visual quality of the result, using different numbers of VALs. The parameter τ defines the percentage of the overall measured radiance, the direct light accounts to. This implies that a low percentage leads to smaller area lights, that concentrate in bright regions while large percentages result in larger areas to be sampled (visualized in the first row). Thereby, higher percentages allow other bright areas to be treated as direct, shadow casting light sources. At the same time, increasing τ also leads to a decreasing intensity, because the area grows while the uniformly distributed average radiance is declining, since only darker texels are added to clusters. For very large τ, like in the bottom row, this leads to wide VALs with relatively low intensity. The effect is even stronger when

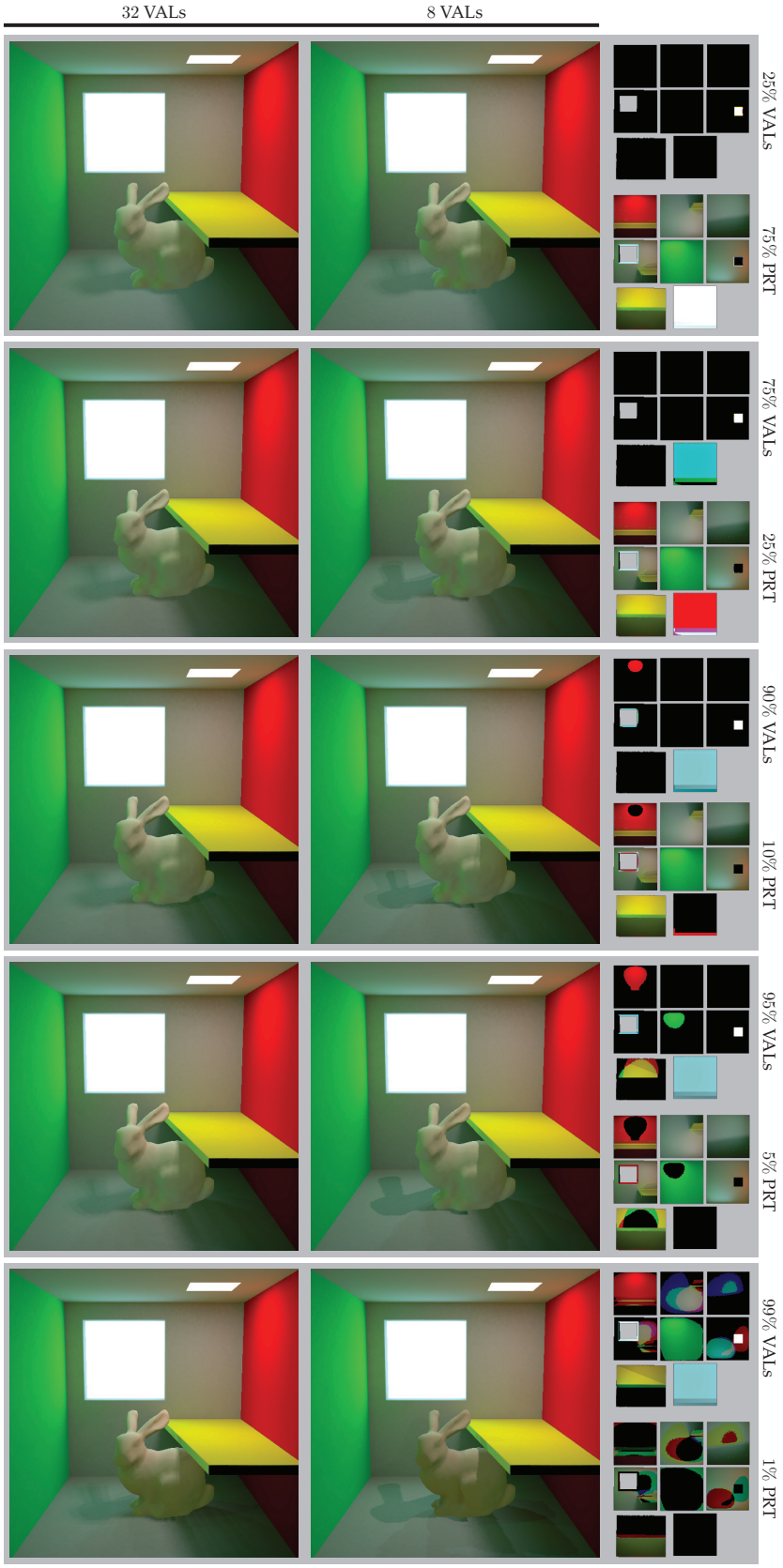


Figure 5.23: Discussion of User Parameter τ

Comparison of different percentages τ to split direct from indirect light and the impact on the final result, using different numbers of VALs.

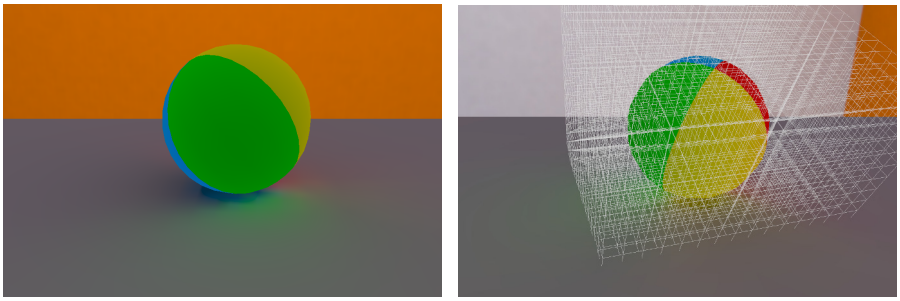
using a low number of VALs, which results in an ambient like shading. Note, that these results differ from the approach of clustered VPLs, discussed in Section 5.4.2. Here, point lights with area-independent intensities are used, which leads to singularities when shading a surface close to the light position. While selecting a very high τ causes problems, a very low percentage can enhance the visual quality, when using a low number of VPLs in a scenario with one or two small bright light sources. In this case, the VPLs have a small area and are placed close to each other, leading to soft shadows (see columns on the left). With increasing τ, the light sources drift apart casting shadows from different directions (see rows at the bottom). In general, one requires more VPLs to achieve soft shadows for higher direct light percentages. In our examples, we use 75-98 % of the total radiant intensity for the direct light.

RESOLUTION OF THE ATLAS In our experiments, the atlas resolution was 1024×1024, which roughly matches the resolution of the cameras, used to capture the environment. If a camera is very close to the real surface, it is possible that valuable information gets lost, because of a lower atlas resolution at that part of the surface. We consider this an unlikely situation, because in real world applications, the cameras will be placed at a certain distance from regions of interest to not distract the users. Furthermore, in digital content creation, it is very common to allocate more space for such interesting regions during the atlas parametrization, which overcomes this concern, at least when the positions of the environment cameras are known. However, if it is required to increase the atlas resolution, the performance impact is restricted to the server.

RESOLUTION OF THE CUBE MAPS FOR INDIRECT LIGHT It is not required to use high resolution cube maps for estimating the indirect light, because the SH projection does not preserve details. If there are fine bright details with visible impact, that are smaller than one cube map texel, it is very likely that those details should be treated as direct light instead. However, if there are small important details, that are not bright enough to be considered direct radiance, these surfaces have to be very close to the virtual object. In this case, the surface will be represented in a larger area of the cube map and thereby handled with little errors. For our experiments, we use a $32 \times 32 \times 6$ $R32G32B32A32$ cube map. We do not recommend lower resolutions, since the results can start flickering when moving or rotating the objects. Using higher resolutions showed no improvements.

5.6 GRID-BASED INDIRECT ILLUMINATION

A limitation of the described approach is the lack of indirect illumination, which is reflected from the virtual object to the real scene. At this point, we ignore that type of light paths, but extensions to overcome this limitation are possible. E.g., by analyzing the radiance distribution on the virtual object and the placement of virtual light sources onto the virtual object. PRT can also be used to achieve low frequency indirect illumination. Figure 5.24 shows the results of experiments to investigate this idea. Here, the PRT framework is used to store the distribution of reflected environment light in a dense volume around the virtual object. At each grid cell, we store the amount of incident reflected light in SH basis and also the visibility at the grid position in SH, both computed in a pre-process. During run-time, the coefficients are



(a) Indirect Illumination (b) Visualization of the Grid

Figure 5.24: Grid-based Indirect Illumination
Experimental results of a grid-based indirect illumination from virtual objects, in this case the colored ball, to an receiver surface based on PRT. The direct illumination of the receiver surface is also computed using PRT, including shadows cast by the colored ball. Note, that the information stored in the grid (b) can be extrapolated. The indirect illumination is increased by factor 3 for illustration purposes.

fetched, based on the position of intersecting receiver surfaces. Using another lookup table allows to fetch receiver coefficients, based on the current surface normal. To evaluate indirect light, we compute the summed product of the grid, receiver and environment light coefficients in the sense of PRT. The direct light in this example is also computed by PRT, using the receiver, visibility and environment light coefficients. For receiver points outside the grid, the illumination is extrapolated and scaled by the inverse square falloff. A more detailed evaluation, especially in terms of more complex geometry, remains for future work. For diffuse environments however, this grid-based solution shows promising results.

5.7 DISCUSSION

We demonstrate that augmented reality with consistent illumination is possible on current mobile devices at interactive frame rates. To achieve this, we developed a lighting method that shares the computation effort among a stationary PC and the participating mobile device. The amount of data to be exchanged between both is reduced, avoiding a bottleneck in transmission due to limited bandwidth. Multiple mobile devices are supported without additional overhead in terms of lighting calculation and transmission, since the parameters of the light model are valid for all devices and can be broadcasted. We capture the near-field illumination of indoor scenarios with multiple HDR video cameras and use this information for the illumination of the virtual objects. These objects can be moved freely with a consistent illumination at any position and their shading adapts to temporal changes in the incident illumination, even though the sources of light are not visible to the tablet camera. Although our system is designed for diffuse virtual objects, we also introduced a first approximation for a plausible display of glossy materials.

For this project, we placed the HDR cameras manually such, that all relevant regions are visible in at least one of the cameras. If there are still regions not visible to any camera, some of the illumination might be missing. To overcome this problem, we suggest to evaluate a dynamic, tracked HDR camera, that can be moved to such invisible regions. Parts of that idea have been incorporated

in the system described in Chapter 7. However, there is still room for future research, as it enables new possibilities due to additional information (see Section 8.2).

While concentrating on the acquisition of the scene illumination, we assumed that the geometry and material reconstruction is provided. Since portable 3D sensors are available, dynamic capturing of the geometry and materials is also highly relevant and discussed also in the context of the project in Chapter 7.

The method supports manipulation of virtual objects with correct illumination at interactive rates, but the update rates of the direct light sources are lower, since they are only updated after a *complete* iteration of the server pipeline, as visible in the supplemental material of Appendix A.1. Additionally, we neither predict the VAL positions on the client side nor blend between updated and former VALs, which would both hide this latency.

To improve the shadow quality, softer shadows could be displayed for each area light, similar to [Don+09]. Increasing the number of VALs and thereby the number of overlapping shadows is another option, which is discussed in the next chapter.

Note, that our hardware setup allows for working in a dynamic environment with moving real objects and under changing light conditions. For static environments, one can evaluate the server pipeline in a pre-process and transfer light positions and the indirect radiance atlas to all participating clients. The clients need to perform the SH compressed indirect light on their own, every time a virtual object has moved, but eventually the presented method will also work in this setting. When using an inside-out tracking (see Section 3.3), the complex hardware setup will not be required at all.

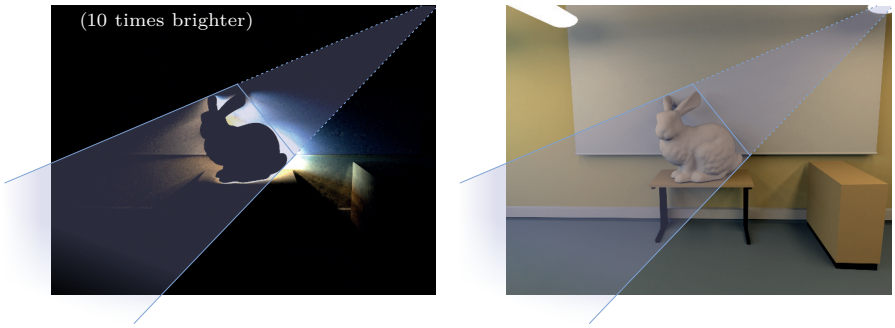
6

TILED FRUSTUM CULLING FOR DIFFERENTIAL RENDERING

This chapter contains an extension of the framework described before. The project and the results have also been presented at a conference and in the corresponding proceedings [RG15]. For this thesis, additional illustrations and further details as well as possible areas of application have been added.

In the previous chapter, we showed how to achieve interactive coherent augmentations by distributing the computations between a stationary PC and mobile devices. To reach real-time frame rates on the mobile device, the number of extracted light sources must be low, limiting the scope of possible illumination scenarios and the quality of shadows. In this chapter, we show how to reduce the computational cost per light, using a combination of tile-based rendering and frustum culling techniques, tailored for AR applications. The approach runs entirely on the GPU and does not require any pre-computation. Without reducing the quality of displayed images, we achieve up to 2.2× speedup, compared to the previous implementation. Even though, the technique was motivated by a potential performance gain, a culling strategy aiming for more general application was developed. The suggested strategy can be used for different point light-based differential techniques and also in the context of VR, where existing video footage or environments with baked illumination need to be extended by interactive virtual content[27].

27 Technicolor. *Immersive Lab*. Website, 2016. Demo was presented at ISMAR 2017 VR Tour.



(a) Absolute Difference (b) Augmented Image

Figure 6.1: Influence of a Single Light Source
Visualization of the influence region of a single light source in the difference image (a) and the final augmentation (b).

6.1 MOTIVATION

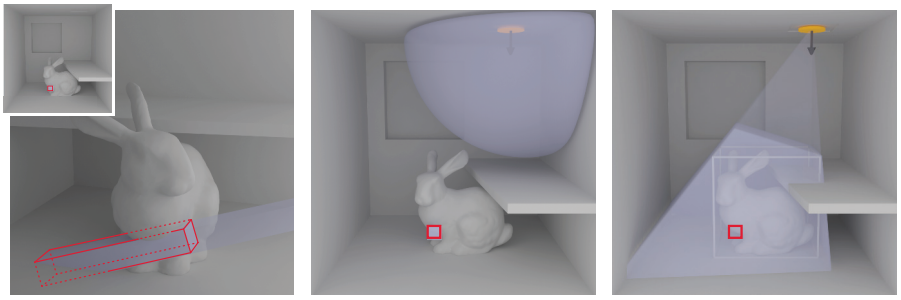
Differential rendering, as described in Section 4.2, is the basic principle of most AR rendering techniques. As described earlier, we compute the per-pixel-difference of two simulations, which can be considered as the influence of the virtual object on the real scene. As radiance is additive, we could also compute a per-pixel difference for each reconstructed light source and accumulate them afterwards. Technically, we could break this down to partial solid angles of incident radiance at each real-world surface, visible in the camera image. However, it is more useful to consider discrete light sources, especially when it comes to real-time applications. Many approaches, starting with GIBSON and MURTA [GM00], represent the real-world light sources by a set of discrete directional or point lights. The distributed illumination framework discussed in Chapter 5 is another example. There, we applied Equation (5.11) to evaluating the total per-pixel difference directly. Investigating the contribution of a single light source (here a VAL), we consider only the case in which the pixel shows no virtual object:

$$\Delta L_{VAL_j}(\mathbf{x}, \boldsymbol{\omega}_o) = -L_{VAL_j}(\mathbf{x}, \boldsymbol{\omega}_o)\ \left(1 - \hat{V}_j(\mathbf{x}, \boldsymbol{\omega})\right)\ V_j(\mathbf{x}, \boldsymbol{\omega})\,.$$

Here, the visibility at $\mathbf{x}$ from VAL j in the reconstructed scene is denoted as $V_j(\mathbf{x}_v, \boldsymbol{\omega})$ and in the virtual scene as $\hat{V}_j(\mathbf{x}_v, \boldsymbol{\omega})$. With $L_{VAL_j}(\mathbf{x}, \boldsymbol{\omega}_o)$ being the non occluded radiance from light j, we subtract that radiance from the background if and only if $\mathbf{x}$ is in shadow of a virtual object and not already in the shadow of a real-world object. Hence, $\Delta L_{VAL_j}(\mathbf{x}, \boldsymbol{\omega}_o)$ can be different from zero, if and only if $\mathbf{x}$ is in the shadow volume spanned by the light j and the virtual object. Figure 6.1 illustrates one of these volumes in the difference image from the example in Figure 4.7.

If the simulation is focused on direct lighting without color bleeding from virtual to real objects, the estimated influence range is correct, but the observation holds even for indirect illumination, too. Considering the instant radiosity technique [Kel97, Kne+10] (also see Section 2.7.2), indirect light is also modeled by discrete light sources. In this case, the influence of secondary sources, is also limited to the shadow volumes, which they are spanning.

Identifying these rather local regions in screen-space allows to discard light transport computations for the rest of the image. This chapter focuses on the detection of the regions without introducing bias to the final image and



(a) Tile Frustum (b) Light Hemisphere (c) Light Frustum

Figure 6.2: Culling Strategies
A screen tile defines a small view frustum – both visualized in red – with depth bounds based on the smallest and largest occurring depth (a). The shading inside the tile depends on lights with influence in this region. Considering only the orientation of a light, all tiles in the hemisphere of that light will compute the contribution of that source during rendering (b). In AR applications and differential rendering the light frustum allows more efficient culling of lights without impact (c).

thereby the improvement of the rendering performance of the method presented in Chapter 5, which allows to compute more precise solutions by increasing the number of processed lights or to account for more complex materials on virtual objects.

6.2 CULLING IN TILED DEFERRED RENDERING

First, let us recap the tiled rendering, described in Section 5.3. The G-Buffer is subdivided into small fixed size tiles, e.g., 16×16 or 32×32 pixels. By taking advantage of the general purpose computing capabilities of modern GPUs, a group of threads can be dispatched for each individual tile. The threads read the G-Buffer and determine the minimum and maximum occurring depth within the tile leading to a small frustum (see Figure 6.2a). Then, the threads test this frustum for intersection with the lights' influence regions in parallel and add intersecting lights to a shared list. For that, each thread handles one light source. When all lights are processed, the list contains relevant lights only and the fragments within the tile can coherently operate on the same list. Now, the role of threads changes. Each thread is now processing one pixel and iterates over the list of the tile to accumulate the radiance for that specific pixel. The threads of different tiles perform the same computations for their input independently.

For the rendering as presented in Section 5.3, the intersection test ignores lights only if the tile frustum is on the backside of a disk-shaped area light (see Figure 6.2b). More efficient tests can help to reduce the number of lights used for shading without influencing the result. Therefore, we test the tile frustum for intersection with the volume spanned by the light position and the bounding volumes of virtual objects (Figure 6.2c), the so called *Light Frustum*. Approximating the object by its bounding volume, e.g., a box, that introduces no errors, as long as the object is entire contained in the approximation. On the contrary, the bounding box is a weaker criteria for rejecting regions of zero contribution than a classical detailed shadow volume [Cro77]. However, assuming that shadow mapping is used for visibility tests, the light frustum

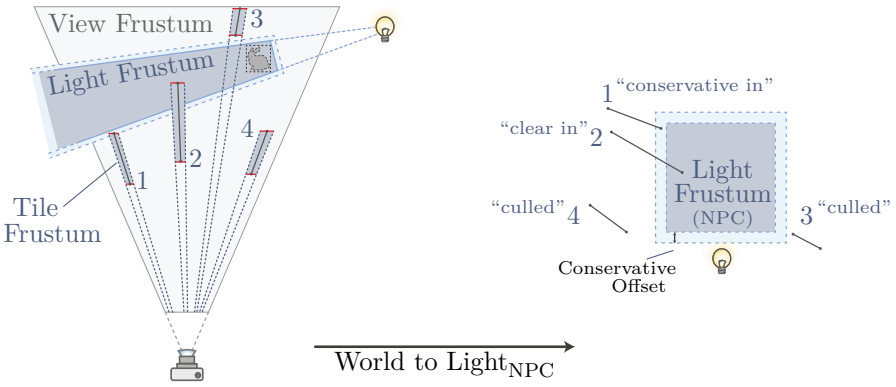


Figure 6.3: Culling for Tiled Deferred Differential Rendering
Intersection tests between tile frustum and shadow volume for deferred rendering. The center point of front and back of each tile frustum is transformed into light clipping space. The resulting line segment is tested for intersection with an axis-aligned box.

based on the box is already given. If all shadow maps are set up to closely fit the virtual objects for achieving optimal shadows with a certain resolution, the view-projection matrix used for the shadow map generation defines the light frustums for each light, too.

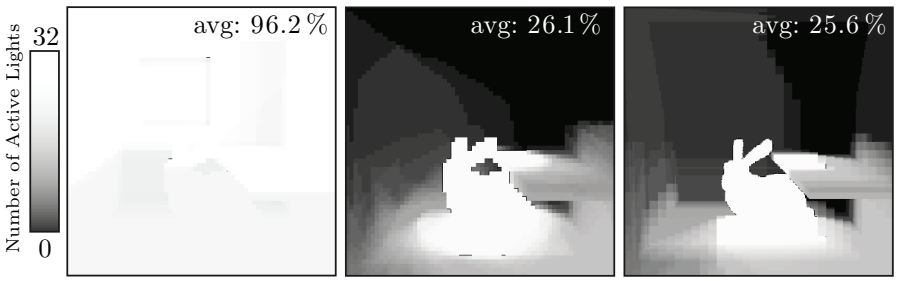
Note, that the hemisphere is actually one with infinite radius and thereby a half-space. The approaches that introduced tiled rendering and the variations of it are focused on performance in games. To speed up their rendering, they keep the lists of relevant lights short by limiting the influence radius of the individual sources. To avoid bias, we do not want to limit the radius. For the light frustum we need to make sure that the far plane distance is large enough to not cut off real-world geometry.

To test if a tile intersects such a volume, a ray through the center of the tile is transformed into *clipping space* (also known as Normalized Projection Coordinates (NPC)) of the light and a simple line-box intersection test can be used to decide about the relevance of the light for illumination (see Figure 6.3).

To prevent false negatives because of the extents of a tile, the size of the box is slightly increased. The *conservative offset* can be applied on the fly for each tile by extending and moving the near-plane of each tile frustum by the diagonal of its backside (the largest extent). The offset can also be incorporated during the setup of the light shadow frustums on the CPU. The corners of the bounding box of an object are projected into screen space, moved by the size of a tile and then the back projected positions are used to fit the shadow frustum.

Figure 6.3 shows this test for four examples. While tile 3 and 4 can be rejected, tile 1 and 2 intersect with the unit box or the conservatively extended box. Figure 6.4b shows the number of lights to process per tile and thereby the result of the culling in the test scene. The impact on the rendering performance will be discussed later in Section 6.4.1.

This technique accelerates deferred rendering without losing accuracy, which allows to use more lights and thereby achieves higher quality. Note, that increasing the number of lights leads to softer shadows because technically they are overlapping hard shadows cast by point lights. Unfortunately,



(a) Hemisphere Test (b) NPC Culling (c) Tiled Light Lists

Figure 6.4: Culling Results Using Different Strategies
Hemispherical test of Section 5.3 (a), NPC culling introduced in Section 6.2 (b), implicit culling using Tiled Light Lists introduced in Section 6.3 (c).

this method is limited to opaque objects because the G-Buffer only contains geometry information of the front-most surface. To enable transparent materials, one could add further layers, leading to increasing memory usage and processing time. Typically, transparent objects are handled separately by forward rendering. As an alternative, *Tiled Forward Shading* was introduced, which combines the benefits of tile-based lighting and forward shading (transparency and material complexity).

6.3 Culling in Tiled Forward Rendering

Traditional forward rendering is still the most common approach in AR and other real-time applications. Assuming modern hardware, light culling is still difficult. Relevant lights have to be identified on a per-fragment level.

Olsson and Assarsson [OA11] introduced tile-based shading to overcome this problem (see Section 2.7.1). Before rendering the objects of the scene, the screen is split into tiles again. Each tile implicitly defines a small view frustum, making it possible to create a per-tile list of lights with potential influence in this tile frustum. During forward rendering, the fragment program determines the current tile and processes the list of lights for calculating the shading. In contrast to deferred rendering, no additional memory is required for a G-Buffer, lowering bandwidth problems. Additionally, *alpha blending* can be used for transparent objects and *multisampling* is supported to reduce aliasing artifacts.

Motivated by *Clustered Deferred and Forward Shading* by Olsson *et al.* [OBA12a], our goal is to find the minimal set of light sources that has to be processed by fragments within a tile. To implement this, the view frustum is tiled not only in x and y direction of the screen but also in *depth*. In contrast to the clustered shading, we propose a new sparse data structure, the Tiled Light List (TLL), which explicitly stores the minimum and maximum distances for the influence regions of the lights within each tile. Before rendering, this data structure is compacted (Section 6.3.3) to speed up light list iterations during the actual illumination computation (Section 6.3.4).

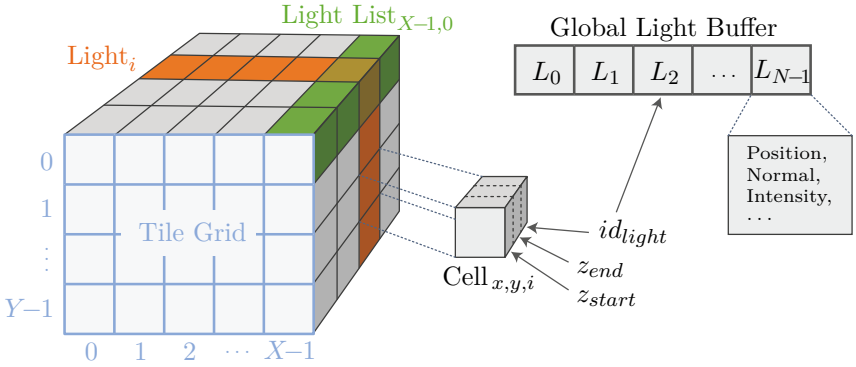


Figure 6.5: The Tiled Light Lists (TLL) - Data Structure
Each screen tile holds a list of nodes. Each node stores the beginning and end of the influence z-range of one light. Since the lists are compacted before rendering, each cell also keeps a reference into the global light buffer that stores all required information per light source.

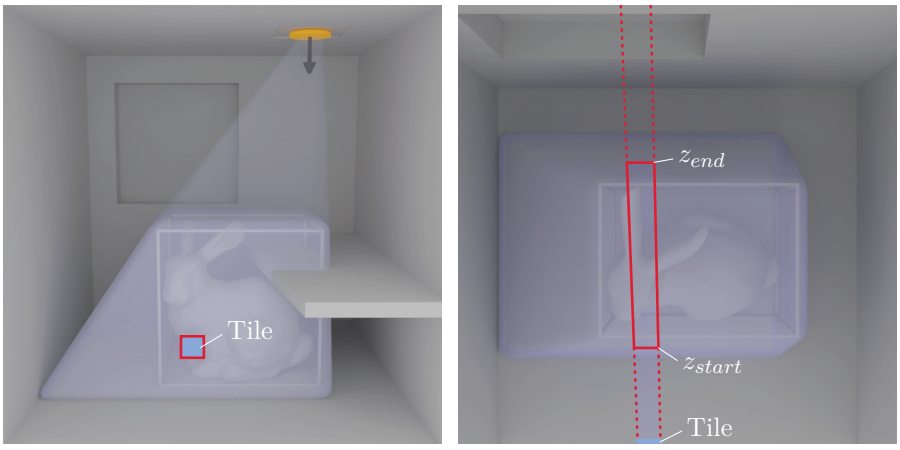
6.3.1 *Tiled Light Lists*

Independent of the renderer used, we assume all N lights to be sequentially stored in a *Global Light Buffer*, containing all required information for each light, i.e., position, direction, area and intensity.

Real-time applications such as games often use a very large number of lights and limit their influence range using a falloff. This reduces the number of lights per pixel, but results in an approximated solution. In contrast, coherent photorealistic rendering requires a *global support* of each light for a correct solution. This leads to a significantly larger footprint of the influence range on the screen. However, for differential rendering we know that only the virtual objects and real surfaces in their shadows will be affected by direct light. Therefore, we will use the volumes spanned by each light source and the bounding boxes of all virtual objects to define the region of potential influence. Since each light in an AR application needs to cast shadows and because of the larger footprint, the number of sources that can be processed is fairly low compared to thousands of small lights in games.

Knowing that the number is limited allows us to allocate memory for each light per tile. Instead of building a three-dimensional grid over the position in the view frustum, we define a 3D grid over the 2D tile coordinates and the number of lights in the third dimension, as visualized in Figure 6.5. The minimum and maximum depth of the influence range for each light, z_{start} and z_{end}, are stored along with the id_{light}, the index into the global light buffer, in the corresponding grid cell. Figure 6.6 shows an example.

We use m bits to store depth values, leading to $2^m - 1$ possible depth values. Because GPUs address memory in blocks of 4 Bytes, m typically is 32, while the smallest and largest numbers are reserved to represent invalid values. The screen resolution is subdivided into squared tiles of fixed size ($k \times k$ pixels). For *FullHD* resolution and typical values ($k = 32$, $N = 64$ lights) this leads to a fairly small amount of memory of about 1.5 MB. For a larger number of lights k can be doubled, leading to a reduction in buffer size of factor 4. If this still exceeds the requirements, more effort can be invested to either pack the data or implement structures like *Fragment Linked Lists* [Yan+10]. However, by using the proposed sparse structure, the light list per tile can be placed



(a) Front View (b) Top View

Figure 6.6: Intersection Between a Tile and a Light Frustum
The frustum that corresponds to a screen-space tile can be intersected with a light frustum (fitted shadow volume). The resulting intersection can be described by z_{start} and z_{end}. During rendering, a fragment is only illuminated by the light, when it is located inside the intersection volume.

linearly in memory to increase cache coherence during the light list iteration. In case of linked list-based implementations, the lists probably need to be sorted to allow evenly fast iterations.

6.3.2 *Filling the Tiled Light Lists*

The focus of this section is the construction of the data structure. We rebuild it every frame and thereby do not need any pre-computation steps and can deal with completely dynamic scenes. For filling the lists, we utilize the rasterization pipeline and further optimize it with the help of GPGPU features.

First, the data structure is cleared to make all lights invalid (illustrated in Figure 6.7a for a horizontal slice of the data structure). This is done by setting z_{start} and z_{end} to a large and small invalid number, respectively. The third value, which is used as either *face counter* or id_{light}, is set to zero.

In the second step, we generate the light frustum for each light source and virtual object. Therefore, we extrude the silhouette edges of the world space bounding box of an object along the direction from the light position virtually to "infinity". To close the volume, caps are rendered on both sides of the extruded faces. The concept of this volume generation is the same as for *traditional shadow volumes* [Cro77], except for the reduction of the mesh complexity for which we only extend the bounding box instead of the object itself. All projections are done on the GPU using geometry shaders and instancing to process the lights in parallel. Graphics APIs like DirectX and OpenGL allow to *disable far plane clipping*, which makes the projection to "infinity" easy to implement. During this process the viewport resolution is set to the size of the tile grid, and the virtual camera is aligned to the mobile camera that captures the background image. The fragment program computes the index in the data structure, depending on the position of the fragment and the id_{light}, passed from the geometry shader stage. The fragments depth z_{frag}, is written into the data structure. Depending on whether the pixel to

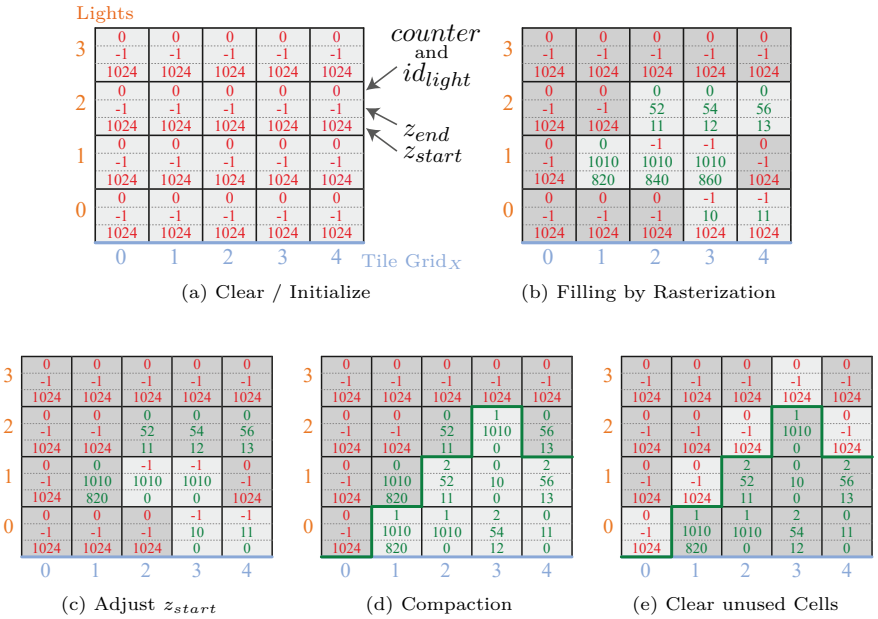


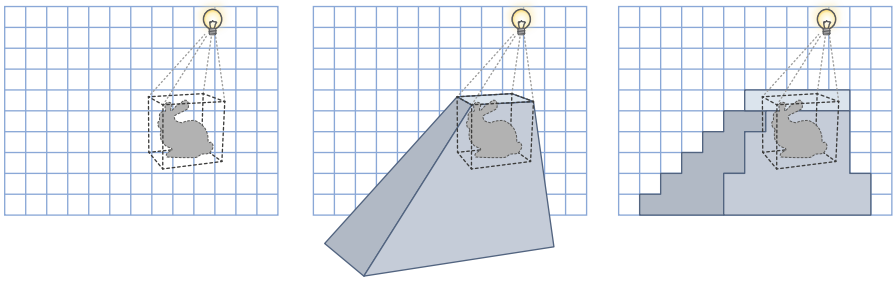
Figure 6.7: TLL Operations

The basic steps of building and updating the Tiled Light Lists. For better visualization, we use only a horizontal slice of the 3D grid (Figure 6.5) with $N = 4$ lights and limit the interval of valid depth values to $z = 0..1023$. Note, that the field of id_{light} is used as *face counter* in step (a), (b) and (c). In the first step (a), z_{start} and z_{end} are initialized with invalid depth values (-1 and 1024), and the face counter is set to zero. In step (b), the light frustums are rasterized conservatively and all affected tiles update their z_{start} and z_{end}. Additionally, the counter is updated (+1 for front faces, -1 for back faces). If the camera is inside a frustum of a light, the counter remains negative. In these cases, z_{start} is set to zero in step (c). In the compaction step (d), the valid lights ($z_{end} \in [0, 1023]$) are packed in a sequential list for each tile. Because the position of the lights are changing, from now on, id_{light} is used to store identifiers for the lights. The lists are filled from bottom to top; due to the parallel insertion, the order of the lights within the list is not deterministic. Finally, the end of each list is marked as invalid in step (e).

process is front or back facing, the depth is used differently. Front faces will update z_{start} to $\min(z_{start}, z_{frag})$ and back faces set $z_{end} = \max(z_{end}, z_{frag})$. The whole process (Figure 6.7b) is repeated for each virtual object in the scene, or bounding volumes enclosing multiple objects. The latter can reduce the computation effort, guided by a hierarchical scene management, e.g., for many small but densely packed objects.

By default the fragment program is invoked only if the center of a pixel is inside the rendered triangle. To account for fragments that slightly overlap the triangle, special treatment is required to ensure that the light list per tile contains an entry also for partly intersecting frustums. This is commonly known as *conservative rasterization* [HAO05]. The spanning of the light frustum and the rasterization is visualized in Figure 6.8.

Similar to traditional shadow volumes it is possible that the camera is located inside a light volume. In this case, z_{start} is invalid after filling the light list, because no front face reached the fragment program. If the volume of the same light and another virtual object intersects the same tile and writes



(a) Object Bounding Box (b) Extruded Bounding Box (c) Conservative Rasterization

Figure 6.8: Rasterizing Light Frustums into TLLs
The bounding box of the virtual object and the processed light (a) span a frustum virtually to "infinity" (b). Using conservative rasterization, all tiles that intersect this frustum can be identified by rendering all extruded faces and the corresponding caps (c).

a valid depth value, the influence range of the light would start at that depth and not at $z_{start} = 0$ as required to contain the first volume. The solution to this problem is similar to classical shadow volumes, too. The id_{light}, that is not used during the filling of the light lists, can be used to count front $(+1)$ and back facing fragments (-1). If more back faces than front faces have been counted, z_{start} has to be set to zero, because at least one light volume contains the camera. In Figure 6.7, this is required for Light_1 (tile 2 and 3 have valid depth values but a negative counter). Light_0 has an invalid z_{start} only and could be handled correctly without counting, but this case is also captured. Note, that it is not possible to count more front than back faces, because far plane clipping has been disabled to guarantee closed volumes at "infinite" depth. As an alternative, two separate variables or the *stencil buffer* can be used for counting similar to classical shadow volumes.

Note, that if a light source is located within the bounding box of a virtual object, the frustum cannot be defined as before. In this case we make this light valid for all screen tiles. At the expense of quality and correctness, it is also possible to limit the range of the lights as frequently done in games. This will speed up shading but introduces bias, which is why we abstained from it in this paper.

6.3.3 *Compacting the Tiled Light Lists*

With regard to later light list iterations, it is desired to have a dense list to be able to abort the iteration as soon as an invalid light is reached.

This compaction can be done in parallel for each tile. A GPU thread group is dispatched for each tile with one thread per light. Each thread reads both depth values and the counter for the corresponding light by using the *thread id* as index into the light list of the tile. A light is considered valid if z_{end} is inside the valid interval. If the counter is negative, z_{start} is set to zero in order to start the influence range of the light right at the camera position (see Figure 6.7c).

The next step is the actual compaction (see Figure 6.7d). Threads with a valid light atomically increment a group-shared counter. The previous value of the counter defines the new target position in the light list of the list. But before writing the values to that position, all threads need to be synchronized

to make sure the data is read into registers before the collaboratively used memory, that stores the list, is overwritten. From this point on, the generic field used for the face counter, now stores the index id_{light} to the global light buffer. Note, that the resulting storage location is a random position in the list because the order of threads incrementing the counter is not deterministic.

Eventually, all threads with an id equal or larger than the current group-shared counter replace the read data with invalid values to avoid lights from being applied twice (see Figure 6.7e).

For a larger number of lights or if the order of the individual lights can be of benefit to the renderer, the compaction can be modified to perform a stream compaction instead [HSO07].

6.3.4 *Using the Lists during Rendering*

Tiled Forward Shading using the TLLs is similar to Forward Shading described by Olsson and Assarsson [OA11]. After updating the TLLs the rendering begins with the opaque objects of the scene (virtual and reconstructed). Each fragment processed calculates the corresponding tile coordinate and depth value z_{frag} based on the position input. All constant terms of the shading model can be gathered in advance, e.g., reading an albedo texture or the shading normal.

The iteration over the light list of a tile is started by reading the first value of z_{start}. If this value is invalid, the iteration can be stopped because no further valid value can be found in the compacted list. In case the current light is valid, z_{frag} is compared to z_{start} and z_{end}. If z_{frag} is lower than z_{start} or larger than z_{end}, the fragment is in front or behind the region of influence and the light can be skipped. If z_{frag} is within the range, the id_{light} is used to gather the required information from the global light buffer and the radiance contribution of the light at the surface is computed and accumulated in the sense of differential rendering, as described by Equation (5.11) in Section 5.3.

A mask to distinguish virtual from real objects is not required anymore, since individual draw calls with specialized shaders can be invoked for reconstructed and virtual geometry. To avoid double shadowing when rendering the reconstructed scene, we again apply two visibility tests: The first one is used to test if the surface is shadowed in the real scene. If this is the case, the light can also be skipped. The second test determines whether the surface is shadowed by virtual objects, which also handles self-shadows while rendering virtual geometry.

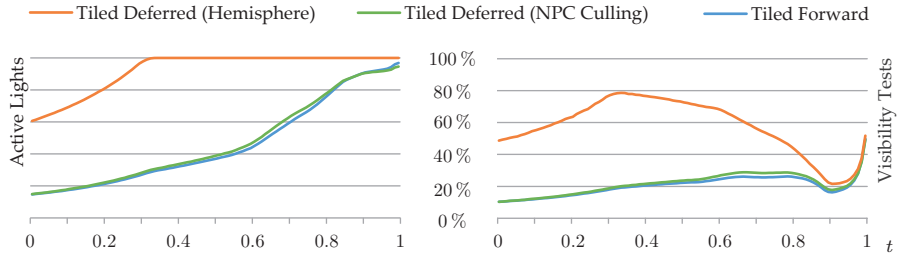
While rendering the reconstructed scene, two radiance values are accumulated simultaneously: one that considers shadows of virtual objects and one that does not. This again allows to compute both light transport simulations in one step (see Section 4.2). Finally, the resulting radiance difference is applied to the background camera image and stored in the backbuffer.

To reduce fragment shader invocations, opaque objects can be sorted front-to-back. It is also possible to do a *z-prepass* and fill the depth buffer in advance. If transparent virtual objects are present, these are sorted back-to-front and then rendered on top using alpha blending. *Order-Independent Transparency* is also possible [Yan+10].

In this section, the resulting performance impact of the proposed culling techniques is reported. All experiments were run on a MICROSOFT SURFACE PRO with INTEL I5-3317U CPU, 1.7 GHz, 4 GB RAM and INTEL HD GRAPHICS 4000. The images are rendered at a resolution of 960×540 pixels and upsampled to *FullHD*. If not stated otherwise, $N = 32$ direct lights are used to represent the environment light. As discussed earlier, the footprint of the influence range of the lights is the most essential factor on the rendering performance (see Figure 6.4). To provide insight in the behavior of the methods under varying magnitude of this footprint, timings have been measured during a simple animation. Figure 6.9 shows three images of this animation sequence, in which the camera moves from a distant viewpoint to a location close to the virtual object. In the first frame, the virtual object covers 11 % of the screen. During the animation this amount increases until no background is visible anymore.

(a) Time $t = 0$ (b) Time $t = 0.4$ (c) Time $t = 0.8$

Figure 6.9: Single Images of the Test Animation
The camera moves to a point near the virtual BUNNY until no background is visible anymore. The black areas left and right of the first image contain neither virtual nor reconstructed objects.



(a) Amount of Lights Processed (b) Required Visibilty Tests

Figure 6.10: Amount of Active Lights and Visibility Tests
The percentage of the lights to process for all pixels (a) shows the efficiency of the culling method. 100 % means all $N = 32$ lights have been evaluated for shading. By checking if a surface position is facing away from the light, the number of required visibility tests (b) and radiance calculations can be (further) reduced for all culling strategies. 100 % means that both visibility tests are been conducted for all $N = 32$ lights.

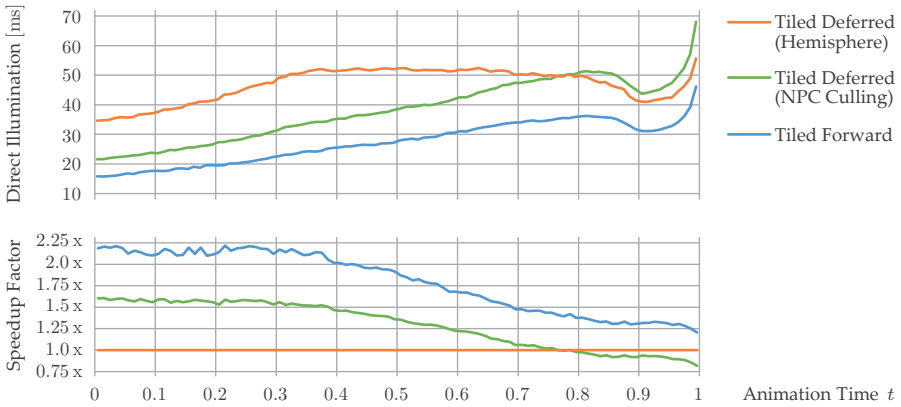


Figure 6.11: Performance on Mobile Hardware

Absolute timings required for rendering the test animation containing a Bunny (top). Performance improvement compared the hemispherical culling (bottom). Timings are measured on mobile hardware (Microsoft Surface Pro) in half resolution (960×540).

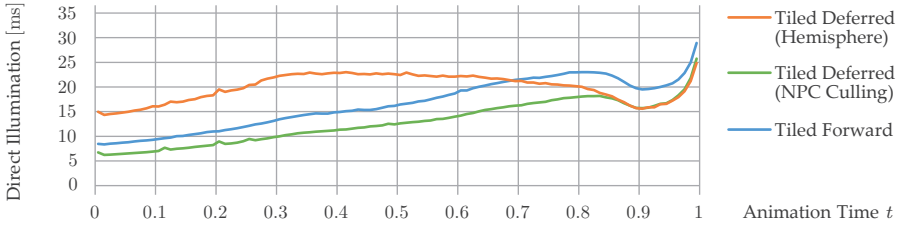


Figure 6.12: Performance on Desktop Hardware

Absolute timings required for rendering the test animation containing a Bunny. Timings are measured on desktop hardware (Nvidia GeForce GTX 580) in full resolution of 1920×1080.

6.4.1 *Culling and Rendering Performance*

The total number of lights considered during each time step is shown in Figure 6.10a. Compared to hemispherical culling, the footprint of active lights has been reduced significantly up to the worst case in which only the virtual object is visible; requiring illumination by all lights. Note, that the curve for the hemispherical culling starts at about 60 % because the scene does not cover the whole screen at the start of the sequence. Figure 6.10b contains the number of visibility tests used for rendering. Naturally, this number depends on the active lights, but the tests can be avoided, if the point to illuminate faces away from the light. Visibility tests are the most expensive operations in terms of computation time. This can be observed by comparing the curves with the absolute timings in Figure 6.11(top). The tiled deferred rendering with the proposed NPC culling technique (Section 6.2) and the tiled forward rendering using TLLs (Section 6.3) show similar numbers of active lights and visibility tests.

The timings however, differ in favor of forward rendering during the whole sequence. This indicates that the heavy use of texture accesses of the deferred rendering is more critical than filling and compacting our data structure, especially on the used mobile hardware. Nevertheless, both culling methods outperform the hemispherical tests up to the point where the virtual object

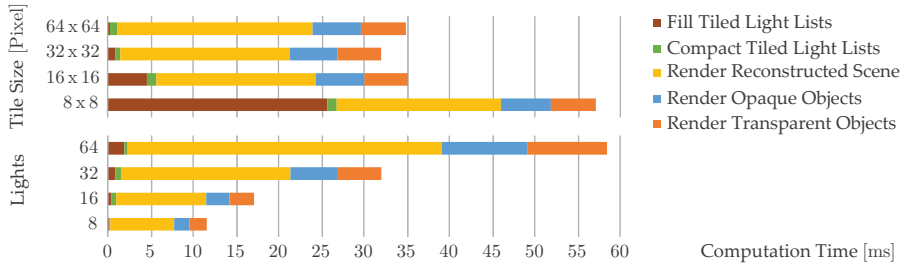


Figure 6.13: Timings with Respect to Tile Size and Light Count
Timing breakdowns of the individual computation steps using Tiled Light Lists depending on the tile size (top) and the maximum number of lights (bottom). The camera position matches the one of $t = 0.4$ of the animation sequence. The scene contains the virtual BUNNY and the first three spheres shown in Figure 6.14. All timings are measured on the mobile device.

covers nearly the entire screen. Figure 6.11(bottom) shows a plot that compares both methods relative to the hemispherical tests. In this worst case scenario both methods introduce a computational overhead because no lights are culled. When applying forward shading, this overhead is lower than the extra time required by both deferred renderers.

Comparing this to measurements of more powerful desktop hardware (see Figure 6.12), shows that the behavior depends on the used graphics hardware. The NVIDIA GEFORCE 580 GTX, used in our tests, does not exhaust the memory bandwidth and the dedicated video memory is large enough to store all buffers. In this scenario, the deferred renderer outperforms the forward rendering, which shows that both approaches are relevant for applications.

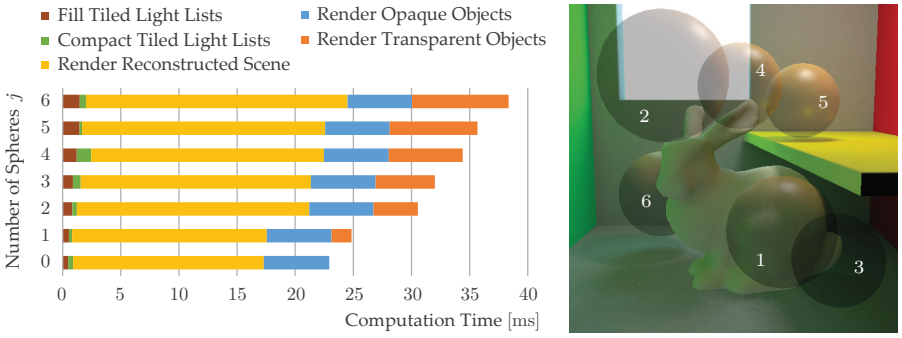
For the deferred rendering, we tested simpler culling schemes, too. For instance, a less complex sphere intersection in NPC of the light instead of the box test, but we found that culling cost amortizes when using more sophisticated culling techniques. This is because the shadow map lookups for visibility tests are more expensive than a few arithmetic operations, especially on mobile hardware.

For all our experiments, we use a tile size of 32×32, but other resolutions are valid options, too. The timing breakdowns in Figure 6.13(top) support this default value because of the lowest overall duration. However, choosing higher or lower tile sizes has influence on both, the preparation of the light lists and the time for iterating them. Reducing the size of the tiles increases the size of the data structure and the rasterization of N light frustums gets the computational bottleneck. Increasing the tile size leads to faster updates of the lists but lowers efficiency in terms of culling because of the conservative generation.

The diagram in Figure 6.13(bottom) shows computation times depending on the number of light sources. The measurements confirm that data structure and forward rendering using this structure scale linearly or slightly better with the number of lights.

6.4.2 *Transparent Virtual Objects*

Aside from a speedup factor up to 2.25, the forward rendering approach also supports alpha blending, as shown in Figure 6.14b. All spheres participate in the generation of the TLL and are illuminated as well as shadowed. The



(a) Varying Object Count (b) Spheres with Indices

Figure 6.14: Amount of Active Lights and Visibility Tests
Augmenting the scene by a Bunny and transparent spheres. Timing breakdowns for different numbers of spheres (a), where j indicates that the spheres with label 1 to j have been rendered (b).

diagram (Figure 6.14a) contains timing breakdowns for the individual computation steps. The required time for the differential rendering of the background and the virtual opaque Bunny are independent of the number of spheres. In contrast to that, the duration needed for filling the list, compaction and rendering of the transparent objects, depend on the size of the spheres on the screen.

Results of real augmented scenes are shown in Figure 6.15 and 6.16. The left side of each figure shows the user's perspective of the real scene without virtual objects and the mobile device displaying the augmented image of the back camera. The images on the right are screenshots of the displayed augmentation, captured by the mobile device.

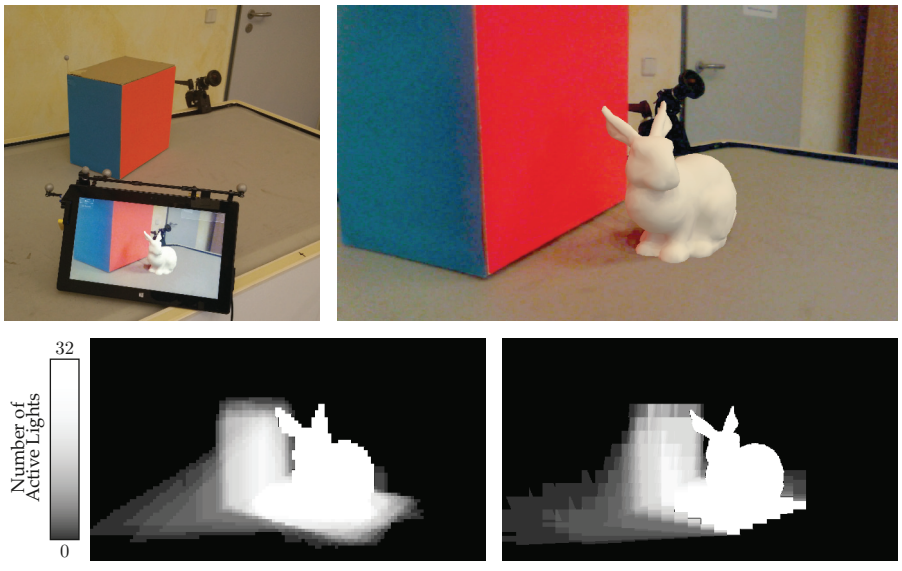


Figure 6.15: Real-world Results for Culling
Mobile real-time augmentation with 32 light sources (top). Simple hemispherical culling (see Chapter 5) can reject no light sources. The improved culling reduces the average number of used light sources per pixel to 21 % for deferred rendering (bottom left) and 20 % for forward rendering (bottom right), leading to significant speedups of factor of 1.5 and 2.

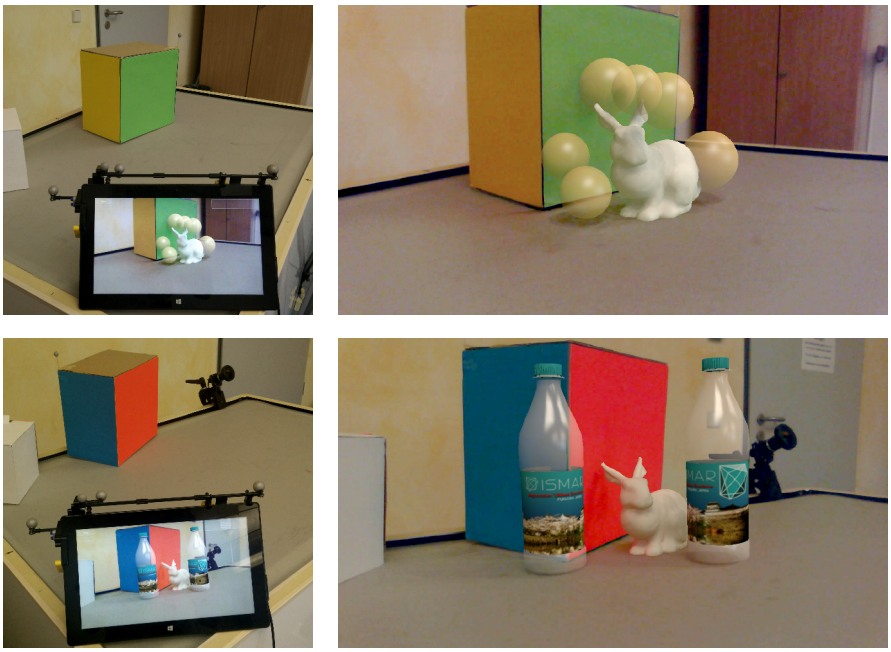


Figure 6.16: Real-world Results with Transparency
Transparent virtual objects in the real camera image. The right-hand side images shows the image that is presented on the device.

All images show virtual objects illuminated by the real environment using $N = 32$ direct lights and PRT for low frequency indirect lighting. The translucent objects in Figure 6.16 are shaded by the same technique as opaque ones (tiled forward rendering), but with alpha blending enabled.

6.5 DISCUSSION

We developed and evaluated two culling techniques for tiled-based differential rendering. The first can be used in deferred rendering whereas the second compiles lists of relevant light source and is thereby more generally applicable.

Both presented culling methods lead to a significant performance improvement when compared to the previous hemispherical rejection strategy. We demonstrated that tile-based light culling is able to increase the speed for differential rendering on a mobile device up to a factor of 2.2 while obtaining exactly the same rendering quality because of conservative rejections. However, if the virtual objects cover nearly the entire screen, the achieved performance benefit decreases because of a computational overhead, that all culling algorithms show in a worst case scenario when nothing is rejected. In most AR scenarios both, the real environment and the virtual information, are of equal importance to the user. Thus, settings in which virtual objects cover only parts of the screen are very typical. These cases benefit from culling especially. To get an impression of the temporal behavior of the system and the full sequence corresponding to Figure 6.9, please have a look at the supplemental material in Appendix A.2.

If a light source is located inside the bounding volume of a virtual object, our methods are not able to define a culling frustum and culling has to be disabled for this particular light. This is also a problem for standard shadow

mapping and could be addressed in the future. A straightforward solution is to test for intersection with a bounding box hierarchy and repeat the filling of the TLL for parts of the objects. However, in case the source is located close to the surface of the model, the problem is only shifted, eventually to a triangle-based processing.

In the future, culling can still be improved by fitting tighter bounding volumes than boxes to further reduce the number of lights required for differential rendering. For glossy materials, lights can be rejected during the shading of virtual objects, too. Only light sources within the cone around the reflected view ray are influencing the shading – assuming that discarding the other direction, in which the BRDF will reflect only a marginal amount of the incident light, is not noticeable by the user. Eventually, more complex scenes can benefit from further constraints on the light volumes. Additional scene information, used for occlusion culling, like *Potentially Visible Sets*, can be used to limit the influence range by setting a specific far plane distance, instead of projecting to "infinity".

Switching to forward rendering extends the *Distributed Illumination Framework* by the support of translucent materials and alpha blending as well as multisampling. The presented technique can be directly applied to the *Differential Instant Radiosity* by KELLER [Kel97], which handles indirect illumination from virtual onto real surfaces, too. These light paths are not yet realized within our framework and thereby a limitation (see Section 5.6).

Especially in mobile games, it is very common to bake complex lighting into textures and to compute shading only for dynamic objects. This is very similar to AR, so these applications could also benefit from the presented culling strategies to reduce their computational effort. The same holds for applications in the near future of VR, where real-world 360° video footage will probably be extended by interactive content.

While it is an open question if TLLs can successfully be applied directly to ray tracing-based methods, we introduce an improved sampling strategy for visibility tests, which is inspired by the presented culling, in the next chapter.

7

NATURAL ENVIRONMENT ILLUMINATION

This chapter summarizes the findings and contributions of a research project, that has also been presented at a conference and in a journal [RJG17]. The idea and the realization of the caching technique, described in Section 7.5.1, was contributed by JOHANNES JENDERSIE, the second author of the article.

Many techniques described in Section 4.3 do not allow for interactivity on mobile hardware. With the framework presented in the previous chapters, we show competitive results on such devices. That system however, requires a complex hardware setup (see Figure 5.2) and a manual reconstruction of the real-world scene. In the discussion of Section 5.5 it was mentioned that the framework can also be applied without a permanent presence of that setup, when assuming that the scene is static and that it can be transferred to the mobile clients. Nevertheless, the scene, including the light condition, must be reconstructed at least once in the beginning. The goal of this new approach is to achieve plausible and compelling results with a mobile device and no other hardware at interactive frame rates. Besides the reconstruction of the real-world surfaces and the coherent rendering of virtual objects based on Monte Carlo sampling, we also focus on the third aspect of visual coherence, the camera simulation (see Section 3.1).

Unlike professional DSLR cameras for which exposure, shutter speed, sensitivity, white balance and other options can be specified, simple mobile cameras lack of these options and often provide only an exposure compensation factor and the ability to select one of a few predefined white balance settings. Mobile sensors and their APIs are designed to allow fast and nice-looking snapshots, but provide no basis for measuring radiance. Even their image processing pipeline and parameters are unknown, so that capturing LDR images, that can be merged into HDR (see Section 3.4), is cumbersome. To overcome this limitation, we present an approach to estimate these unknown adjustments with respect to a reference image. This estimation is successively applied to new LDR images to transform them into the reference space for further processing. In the context of AR, we use the approach to create a reconstruction of the environment and to superimpose live camera images by virtual objects, using one of multiple proposed differential rendering variants. Since the manufacturers of the cameras applied all the automatic adjustment in the first place to provide the best possible looking pictures, it is reasonable to transform our augmented image back into the original input color space and to assume, that the color settings are also valid for the augmented scene. Therefore, we eventually transform the composed image back into the original color space of the input image by applying the inverse of the previously estimated transformation.

Figure 7.1: Results of Natural Environment Illumination
A Toy Car with small metal applications rendered, using different quality settings: *low* (left), *high* (center) and *medium* (right). The last image (bottom right) shows the automatic adaption to changing camera parameters while pointing the device towards the bright windows. Model courtesy of Vilac.

7.1 Overview

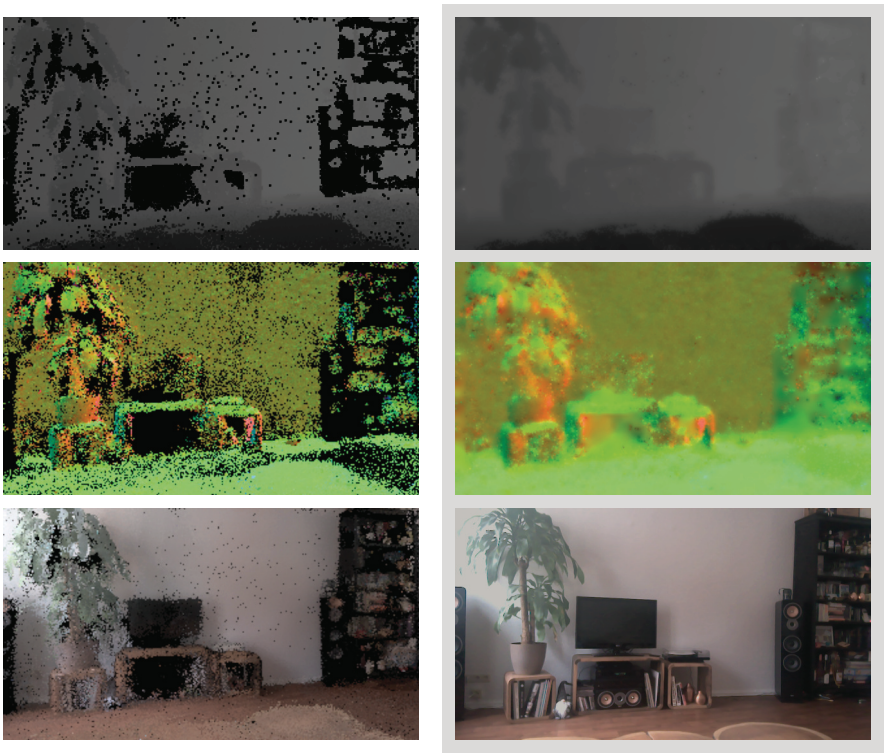
We are using the Google Tango Development Kit[28] which provides an LDR image, a 3D pose estimation of the camera (see Section 3.3.2), and a sparse depth image as well as timestamps for each of them. Based on this input the following steps are performed:

Color compensation estimation To cope with *unknown color compensation* applied by mobile camera drivers, we estimate a compensation with respect to the already captured scene (see Section 7.2). The previous captured parts are rendered from the new camera position and the resulting *Reference Image* is compared to the *Current Input Image*. Pixels that probably show the same real-world geometry are selected and used in a system of equations to estimate the *Color Transformation* that maps the color of the input pixels to the color of the reference pixels. This transformation is applied to all pixels of the most recent LDR input image, which yields a *Corrected Input Image* for further processing and for augmentations later.

Capturing the environment The captured scene is stored as an unstructured point cloud, which is updated every time a new set of samples is provided by the depth sensor. These samples are back-projected from image space into 3D world space and the resulting position is stored with further information, like corrected color and estimated normal. We call these small structs *Environment Samples* and store them in a probabilistic hashmap [HJ10]. The normal estimation as well as the insertion and deletion of samples are detailed in Section 7.3.

G-Buffer as AR basis To allow for basic occlusion between virtual and real objects a z-buffer, containing the real-world objects is required. *Splatting* the point cloud into screen space yields the z-buffer but also normals and colors – which are actually used as reference image in the color compensation step and for material estimation during rendering. After filling holes by push pull steps [MKC07], we have a valid G-Buffer [ST90] that can be used for

28 Google. *Project Tango*. Developer Website, 2017.



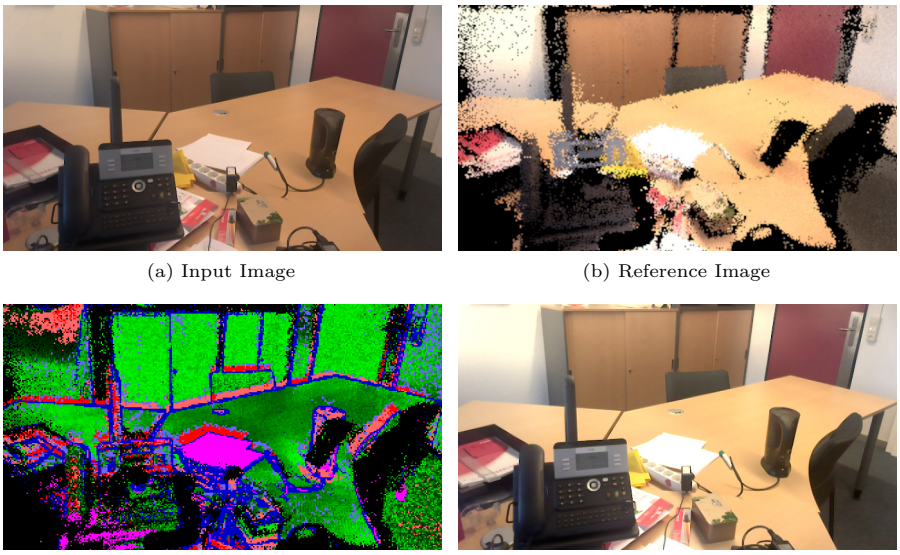
(a) Splatted Environment Samples (b) G-Buffer

Figure 7.2: G-Buffer Generation
Splatted environment sample depths, normals and radiance (a). The G-Buffer, containing depths and normals, filled by push pull, as well as the corrected color image (b). The splatted radiance (bottom left), is not used for rendering but as reference image during the compensation estimation.

a basic and more advanced AR rendering (see Figure 7.2). As the G-Buffer generation is straightforward, we will not provide further details. Note, that we do not need the depth sensor anymore. So, as long as the camera pose is provided, any device can be used. However, depth sensing could be used to improve the G-Buffer in future extensions.

COHERENT RENDERING For the rendering of the virtual objects, we suggest using an image-based ray tracing variant. Therefore, we combine GPU Importance Sampling [CK07] (see Section 2.7.4) with Impostor Tracing [Szi+05]. A Distance Impostor (DI), basically a RGB-Depth environment map, centered around the virtual object, serves as input. This structure is similar to Reflective Shadow Maps [DS05] and is created just like the G-Buffer before. We apply three alternative implementations for differential rendering [Deb98]: the suggested Distance Impostor Tracing (DIT), Environment Mapping (EM) and Voxel Cone Tracing (VCT); for comparison and to emphasize the flexibility of the approach. We also present an improved sampling strategy tailored for differential rendering (see Section 7.4 and 7.5).

INVERSE COLOR COMPENSATION. Since the unknown compensation, estimated in the first step, was introduced by the manufacturer to look pleasant to the user, we transform the augmented image back into the original input image color space, expecting an adequate tone mapping.



(a) Input Image (b) Reference Image

(c) Sample Selection (d) Corrected Image

Figure 7.3: Color Compensation
A camera input image in an unknown color space (a) and the corresponding reference image created by splatting the point cloud (b). The color-coded selection of sample pairs (c) and the corrected input in reference color space (d). Note, that this process produces HDR values, outside the $[0, 1]$ range.

7.2 COLOR COMPENSATION ESTIMATION

Assuming we already captured parts of the environment and these parts are at least partially visible in the current camera image, we aim for estimating a transformation to map between the visible colors of the overlapping areas. Figure 7.3 illustrates the basic idea. The inputs at time t are a camera image in an unknown color space and a rendering of the partially acquired scene in the reference color space. We select pairs of pixels from the images that probably show the same real-world geometry. The pixels selected from the input image form the set $\mathcal{U}_t = \{\mathbf{u}_i\}$ and pixels from the reference image the set $\mathcal{R}_{t-1} = \{\mathbf{r}_i\}$, respectively. After estimating the compensation function c_t that maps the color of the pixels in set $\mathcal{U}_t$ to the colors in $\mathcal{R}_{t-1}$, we can apply the function to all pixels of the input image to transform them into the reference color space. Additionally, we obtain the inverse function to map augmented images back into the original camera color space:

$$c_t : \mathcal{U}_t \to \mathcal{R}_{t-1} \qquad\qquad c_t^{-1} : \mathcal{R}_{t-1} \to \mathcal{U}_t.$$

7.2.1 *Selecting Sample Pairs*

The selection of pixels, that are used in the linear system decides about the robustness of the result. On one hand, we need a large set of samples to get a good estimation. On the other hand, we need to avoid wrong pairs, for instance based on occlusion of not yet scanned geometry. In the first step, we remove all pixels that are easy to reject. This includes pixels that:

- are under or over-saturated in the camera image,
- have no data in the rendering of the point cloud,
- have strong color gradients in either of the images.

Strong gradients are likely to contain edges that could be matched to the wrong surface because of inaccurate tracking.

In the next filtering step, the pixel colors are transformed into Hue-Saturation-Value (HSV) color space. After mapping hue and saturation into $[0, 1]$, we reject pairs that show a very strong discrepancy in color, based on a coarse filter applied on the Euclidean distance between $hsv(\mathbf{u}_i)$ and $hsv(\mathbf{r}_i)$ in the hue-saturation-plane $(\cdot)_{hs}$. Note, that the brightness value has no influence here.

Assuming that changes in the input are small between two frames, we can transform the input set $\mathcal{U}_t$ by the estimated function of the last frame c_{t-1}. Hence, $c_{t-1}(\mathbf{u}_i)$ will be close to $\mathbf{r}_i$ and stronger filtering parameters can be used than before. This time, we reject pairs based on their Euclidean distance in HSV space including the brightness dimension. Figure 7.3 shows the input, the output and a color coding of the filtering process. Green pixels are accepted with a certainty weight w_i according to their brightness, see Equation (7.1). Blue ones are discarded because of gradients, purple pixels were rejected due to saturation and the red ones have a too large distance in HSV.

At this point, different models can be fitted to estimate c_t. We decided to use a *linear* transformation in linear RGB space to describe c_t. We also fitted a coarse *Approximation* a_t that simply aligns the average intensity per color channel. It serves as backup in case of failure, e.g., because of too few samples, and during regularization, which is explained below. Depending on the sensor, other color spaces and models might work better.

We choose a linear transformation here because we cannot make assumptions about the actual type of correction. However, we want to find a robust estimation that can deal with basic operations like a scale in brightness, small color shifts introduced by changing white balance and simple contrast enhancements. To allow any of these operations, the input colors need to be in a linear color space. Ideally, this requires a transformation of the real input, using the camera response curve which is unknown, too. Debevec and Malik showed how to reconstruct such a camera curve for an exposure series [DM97]. Unfortunately, it is not clear how to reconstruct a camera response curve in a system without exact control over the camera parameters. Based on the measurements in [ZCC16], we achieved good results by assuming a default gamma curve of 2.2. For more details on this topic see Section 3.4 on radiometric calibration.

7.2.2 *Color Map Approximation*

For both, the approximation and the linear estimation, the samples are weighted based on their similarity in the hue-saturation-plane:

$$w_i = \left(1 - \min\left(1, \left\|hsv(\mathbf{u}_i)_{hs} - hsv(\mathbf{r}_i)_{hs}\right\|_2\right)\right)^{\beta}. \tag{7.1}$$

The exponent β allows to further penalize larger distances. In all our scenes, $\beta = 2$ was chosen for a_t and $\beta = 8$ for c_t, which showed good results in the experiments. To get a coarse approximation of the unknown color compensation, an average scaling per color channel, $\mathbf{s}$, is computed:

$$\mathbf{s} = \frac{\overline{\mathbf{r}}}{\overline{\mathbf{u}}} = \frac{\sum_i^n \mathbf{r}_i \cdot w_i}{\sum_i^n \mathbf{u}_i \cdot w_i},$$

where n is the number of pixel pairs. The per color channel scaling results in the approximation matrix A_t:

$$A_t = \begin{bmatrix} \mathbf{s}_r & 0 & 0 \\ 0 & \mathbf{s}_g & 0 \\ 0 & 0 & \mathbf{s}_b \end{bmatrix}.$$

7.2.3 *Linear Map Estimation*

To setup the linear system, the sample sets are written as matrices. The estimation function c_t is described as a linear matrix C_t:

$$U_t = \begin{bmatrix} \mathbf{u}_{1_r} & \mathbf{u}_{2_r} & \dots & \mathbf{u}_{n_r} \\ \mathbf{u}_{1_g} & \mathbf{u}_{2_g} & \dots & \mathbf{u}_{n_g} \\ \mathbf{u}_{1_b} & \mathbf{u}_{2_b} & \dots & \mathbf{u}_{n_b} \end{bmatrix} \text{ with } \mathbf{u}_i \in \mathcal{U}_t$$

$$R_{t-1} = \begin{bmatrix} \mathbf{r}_{1_r} & \mathbf{r}_{2_r} & \dots & \mathbf{r}_{n_r} \\ \mathbf{r}_{1_g} & \mathbf{r}_{2_g} & \dots & \mathbf{r}_{n_g} \\ \mathbf{r}_{1_b} & \mathbf{r}_{2_b} & \dots & \mathbf{r}_{n_b} \end{bmatrix} \text{ with } \mathbf{r}_i \in \mathcal{R}_{t-1}$$

$$C_t \cdot U_t = R_{t-1}.$$

To solve for the compensation matrix C_t, we minimize the following energy function:

$$\arg\min_{C_t} \quad \|W_t \left(C_t \cdot U_t - R_{t-1}\right)\|_2 ,$$

where the diagonal matrix $W \in \mathbb{R}^{n \times n}$ at time t holds the weights of the sample pairs $W_{ii} = \sqrt{w_i}$ computed by Equation (7.1). Solving in the sense of least squares yields:

$$\begin{aligned} & \left(C_t\, U_t - R_{t-1}\right) W_t^2 \left(C_t\, U_t - R_{t-1}\right)^T \\ = \quad & \left(C_t\, U_t - R_{t-1}\right) W_t^2 \left(U_t^T C_t^T - R_{t-1}^T\right) \\ = \quad & C_t\, U_t\, W_t^2\, U_t^T C_t^T - 2\, C_t\, U_t\, W_t^2\, R_{t-1}^T - R_{t-1} W_t^2\, R_{t-1}^T. \end{aligned} \tag{7.2}$$

Setting the derivative of Equation (7.2) with respect to C_t to zero, gives:

$$\begin{aligned} & 0 = 2\, U_t\, W_t^2\, U_t^T C_t^T - 2\, U_t\, W_t^2\, R_{t-1}^T \\ \Rightarrow \quad & C_t^T = \left(U_t\, W_t^2\, U_t^T\right)^{-1} U_t\, W_t^2\, R_{t-1}^T \\ \Rightarrow \quad & C_t = \left(\left(U_t\, W_t^2\, U_t^T\right)^{-1} \left(U_t\, W_t^2\, R_{t-1}^T\right)\right)^T. \end{aligned} \tag{7.3}$$

Further, we can incorporate knowledge about the expected transformation. We know that the change in exposure is a strong influence factor that basically scales the brightness, and this scaling factor is already computed by the approximation A_t. To improve the robustness, a regularization can be applied to force the solution closer towards A_t. Therefore, the energy to minimize is changed to:

$$\arg\min_{C_t} \quad \|W_t \left(C_t \cdot U_t - R_{t-1}\right)\|_2 + \gamma \left\|C_t - A_t\right\|_2 ,$$

Now, solving in the sense of least squares, following the scheme above, yields:

$$C_t = \left(\left(U_t \cdot W_t^2 \cdot U_t^T + \gamma I\right)^{-1} \cdot \left(U_t \cdot W_t^2 \cdot R_{t-1}^T + \gamma A_t\right)\right)^T, \tag{7.4}$$

where I is the identity matrix and γ a small scalar factor, set to $5 \cdot 10^{-2}$ in all our examples.

Note, that both matrices in the inner braces of Equation (7.4) are of size 3×3 so they can easily be computed on the GPU. In fact, the entire process of estimating A_t and C_t requires only one iteration over all sample pairs and a parallel reduce operation to compute the sums and the matrix products.

For a large number of sample pairs, usually $n > 2000$ in our experiments, and input images that do not lack information in one or more channels completely, the system is non-singular and can be inverted. This is important to transform rendering results back into the original color space of the input image, as described in Section 7.4.6.

If the sample count is below this threshold, the approximation A_t is used and the frames are marked as approximated to skip the capturing of environment samples for these frames until the compensation is found again. In the first frame we initialize the system by using the identity, $C_0 = I$, which renders the color space of the first frame, the reference color space. While moving in the scene, darker and brighter areas are observed with automatically selected appropriate camera settings. Hence, floating point precision images of different levels of brightness are provided for further HDR processing.

This approach is applicable to any camera system, independent of the availability of a depth sensor. However, depth information could be used in pair filtering.

7.3 CAPTURING THE ENVIRONMENT

The goal of this stage of the pipeline is to create a reconstruction of the real environment. For basic augmentations, the position of the real surface is sufficient, whereas for coherent rendering, normals are required as well. As input serve the sparse depth image delivered by the sensor (see Figure 7.4(left), gray dots on top of the color image) and the corrected camera image to gather corresponding color values. Each measured sample is back-projected to world space coordinates and stored in a buffer. Its buffer index is denoted as unique sample id i. Normals could be approximated by computing the gradient of the interpolated positions. To focus on smooth normals, we address the problem by fitting a plane into the samples of the local neighborhood of each pixel in the sparse depth image.

Figure 7.4 illustrates this process step by step. First, we identify the samples of the local neighborhood by using the rasterization pipeline of the GPU. Therefore, we span a disk around each sample. The radius depends on the

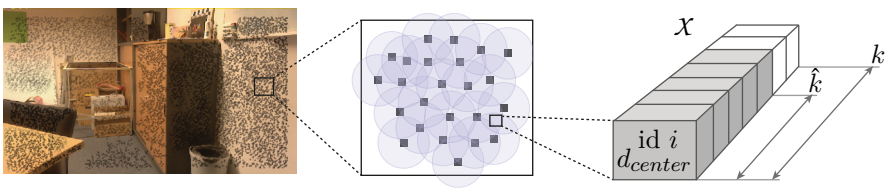


Figure 7.4: Normal Estimation Pipeline
Sparse depth sample given as input (left) are spanned to overlapping disks (center). All disks that overlap a certain pixel are added into a list $\mathcal{X}$ at that pixel (right). The $\hat{k}$ elements in the list form the local neighborhood of the pixel.

resolution of the depth image and the density of the measured samples. It is a device specific constant that must be specified such that the number of overlaps per pixel equals the size of the desired neighborhood k. For each pixel of the depth image, a list of size k is used to store the IDs of the overlapping disks and thereby the pointers to all samples of the local neighborhood.

In the second step, a plane is fitted to the world space positions in the list of each pixel. We now consider an arbitrary pixel and denote the respective list as set $\mathcal{X}$ containing the samples $\mathbf{x}$. Generally, this list is not entirely filled, so we define the number of samples in the list as $\hat{k} \leq k$. Even though plane fitting in 3D is a common problem, we need to perform this task for each pixel as fast as possible on the GPU. Since it is an eigenvalue and eigenvector problem, the common solution requires a Singular Value Decomposition (SVD), which is not feasible in our context. By reformulating the problem, the solution can be found easier. Like in a *Principal Component Analysis*, we first shift the mean of all samples into the origin of a local coordinate system, so the resulting plane will contain the origin:

$$\hat{\mathbf{x}}_i = \mathbf{x}_i - \frac{1}{\hat{k}} \sum_j \mathbf{x}_j \quad \text{for } i = 1, \ldots, \hat{k}.$$

Each vector $\hat{\mathbf{x}}$ defines a direction from the local origin. The normal $\mathbf{n}$, we are looking for, is perpendicular to that direction:

$$\hat{\mathbf{x}_i}^T \mathbf{n} = x_i \cdot \mathbf{n}_x + y_i \cdot \mathbf{n}_y + z_i \cdot \mathbf{n}_z = 0.$$

The trick to avoid the SVD is to split the problem into three smaller ones that are easy to solve. After shifting by the mean, we are now looking for a plane that intersects the origin. There are two degrees of freedom describing the rotation of the plane, but we have three unknown components of $\mathbf{n}$. We now assume that the normal does not lie in the xy-plane. In that case, we can set $\mathbf{n}_z = 1$, such that $x_i \cdot \mathbf{n}_x + y_i \cdot \mathbf{n}_y = -z_i$ and solve the overdetermined system, using normal equations for the other two components:

$$\begin{bmatrix} \sum x_i x_i & \sum x_i y_i \\ \sum y_i x_i & \sum y_i y_i \end{bmatrix} \begin{bmatrix} \mathbf{n}_x \\ \mathbf{n}_y \end{bmatrix} = - \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \end{bmatrix}.$$

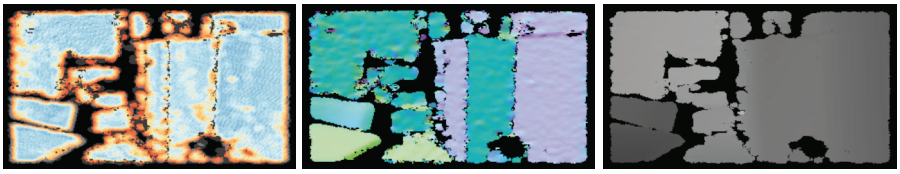
This small system can be solved efficiently by using the determinant and CRAMER's rule, which yields the normal direction $\begin{bmatrix} \mathbf{n}_x & \mathbf{n}_y & 1 \end{bmatrix}^T$, that still has to be normalized.

This does not work, if the normal lies in the xy-plane in which case the determinant gets zero. Therefore, the assumption is altered and the determinant is computed for all three cases. Eventually, only the case with the largest determinant is evaluated to compute the plane normal $\mathbf{n}$. In combination with the mean sample position that was used to shift the samples into the local origin, a world space plane is defined. This approach for fitting planes in 3D is not new[29], but it is well suited for a GPU implementation.

To avoid fitting planes across discontinuities in depth, we search for the sample with the smallest screen space distance to the considered pixel and discard samples, based on their depth distance to this reference sample. In our experiments, a threshold of 0.1 m shows good results even when the surface is seen from a grazing angle.

Intersecting the fitted plane with the view ray from the camera through the considered pixel yields a distance that is used to refine the depth image and thereby the position of the samples.

29 Emil Ernerfeldt. *Fitting a plane to many points in 3D*. Blog, 2015.



(a) Number of Elements (b) Normals (c) Depth

Figure 7.5: Normal Estimation Results
Color coded number of elements used to fit a plane (a), surface normals (b) and improved depth (c).

Figure 7.5 shows the resulting depth and normal image as well as a visualization of $\hat{k}$ for each pixel, where red means a small number and dark blue represents the maximum number. As can be seen, most of the pixels show a medium blue, so the lists are not entirely filled. This is intended, as the radius, that is used during the filling of the list, needs be chosen conservatively. If the radius is too large, it is possible to miss neighbors because the list is already full. In case of two or less samples in a list, we cannot estimate a normal, which is visualized by a dark gray color or black, if there was no sample at all. We also stop the normal computation, if the largest determinant is close to zero, which means that the system is of bad condition. This case is illustrated by a light gray.

After creating the depth and normal image, both are sampled at the locations of the input samples and the gathered information are added as environment sample to the point cloud describing the scene. This point cloud is stored in a probabilistic hash grid with space for 4M elements and a virtual cell size of 0.005 m. If there already is an element at the insertion address, it is replaced randomly with a probability of the inverse number of collisions in that cell. Therefore, a counter per cell keeps track of the collisions.

Besides adding elements to the cloud, samples can be removed, too. This is important to handle movements of real objects and erroneous samples. An old environment sample can be removed, if it has a smaller view space depth than the current depth image at the same location. Obviously, it is not present in the current view and therefore should not exist. The removed samples are often those, which have been captured over a larger distance. Nevertheless, these far samples cannot be ignored in the first place, because they are valuable for generating the reference image.

7.4 INTERACTIVE COHERENT RENDERING

As illustrated in Figure 7.6, our differential rendering system uses an interface to *query* radiance from the scene and from the virtual object. By providing alternative implementations for the environment queries we can offer different quality/performance options. However, we are also able to compare the proposed DIT to the commonly used EM (related to DEBEVEC [Deb98]) and VCT (related to CRASSIN, FRANKE [Cra+11, Fra14a]).

In Section 7.4.1, we describe how the captured point cloud is processed into the data structures, required by the different implementations. We introduce the representation of the virtual object in Section 7.4.2 and show how to use the scene representation to illuminate this object in Section 7.4.3 and Section 7.4.5. To compute the influence of the object onto the environment, we use differential lighting as explained in Section 7.4.4. Details to the final

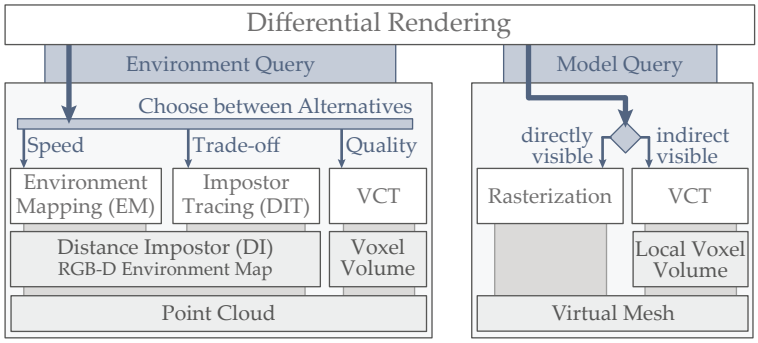


Figure 7.6: Overview of Techniques and Data Structures
The differential rendering queries radiance of the environment and the virtual object. Different implementations can be used to answer the request. In general, such a query can be interpreted as ray or cone cast.

composition are covered in Section 7.4.6. Details on how to solve equations in Section 7.4.3 and 7.4.4 by using Monte Carlo Integration are provided in Section 7.5. Here, we introduce an idea to improve the rendering with only a few samples, by incorporating knowledge of the object influence.

7.4.1 *Scene Representation*

The differential rendering, including the rendering and shading of the virtual object, requires to query the environment illumination multiple times. Each of these queries is a ray or cone cast from the surface position $\mathbf{x}$ into the environment to look up the radiance incident from the direction of the ray, the direction of solid angle $\boldsymbol{\omega}_i$. Instead of dealing with large sets of points, we use dedicated representations for the techniques to realize this query. For EM and DIT this representation is a DI, whereas for VCT a voxel volume is used.

(SCENE) VOXEL VOLUME Cone tracing requires a dense regular voxel volume with a filled mipmap chain [Cra+11, Fra14a]. Therefore, all samples are added to the most detailed mipmap level of the volume where each voxel stores the RGB radiance of the environment samples and an opacity value of 1.0. After generating the mipmap chain, the voxels in the higher levels store averaged radiance and opacity values for lookups with larger cone radii. Assuming a static scene, the volume is filled only once after completing the environment acquisition and contains the entire scene. Depending on the size and the resolution of the volume, this data structure is usually limited due to the memory requirement.

DISTANCE IMPOSTORS To be able to deal with larger scenes we suggest using a distance impostor, which is an environment map that stores radiance and distance into all directions from a specified center position. It can be created by splatting – just like the G-Buffer – while using a cube map as target. The position of the DI should be close to the virtual object – ideally at the center of it. Thus, it needs to be updated as the object is moved in the scene, but depending on the size of the object a threshold can be used to limit the updates to trigger only after larger changes in position. The orientation of the map can be aligned with the world coordinate system for easier access.

To enable ray or cone casting from positions different from the center position we use *Impostor Tracing* presented by Szirmay-Kalos *et al.* [Szi+05]. Here, the direction for the lookup into the map is refined iteratively based on the stored distance to approximate the correct intersection point of the ray with the environment. This process is similar to the secant method, used for approximating the root of a function.

To favor speed over accuracy we also investigate standard environment mapping, which simply uses the color channels of the DI but also depends on the depth channel for occlusion tests.

7.4.2 *Virtual Object Representation*

To compute the shading of virtual objects there are no special requirements and any triangle meshes can be used.

While the virtual object itself is rendered using the rasterization pipeline, additional ray or cone casts are required for indirections to enable self-shadows and reflections for instance (see Figure 7.6).

Independent of the scene representation, we decided to use VCT for this task, as testing the whole triangle mesh for intersections is too expensive. Therefore, we use a solid volume with mipmaps that is small, tightly fit to the object, and is defined in the local coordinate system of the object. The solid voxelization is done on the GPU [ED08] and does not change over time as long as the object is not deformed, e.g., by vertex skinning [KSO10]. Rigged transformations can be applied to the ray or cone instead of rebuilding the volume. However, the meshes need to be closed (watertight), which is a limitation of the solid voxelization algorithm [ED08]. The volume also stores materials and local normals at the boundary voxels.

7.4.3 *Rendering the Virtual Object*

To compute a physically correct shading of the virtual object surface we aim for solving Kajiya's rendering equation:

$$L(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\mathcal{H}_+} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\, L(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i \; \partial\boldsymbol{\omega}_i. \qquad \text{(2.12 revisited)}$$

where the self-emission term $L_e(\mathbf{x}, \boldsymbol{\omega}_o)$ was omitted. The integration domain $\mathcal{H}_+$ is the hemisphere over the surface, as visualized in Figure 7.7a. The figure also shows outgoing sample directions that are concentrated around the reflected view direction ($\boldsymbol{\omega}_o$). This distribution depends on the BRDF, in our case a normalized Blinn-Phong BRDF for glossy materials as described by Hoffman *et al.* [Hof+10], but any other BRDF, that can be sampled properly, might be used instead. To evaluate this equation, we use ray or cone tracing into these reflected directions (see Section 7.5).

To include self-shadowing of the virtual object, each ray is not only traced in the environment but also in the dense volume of the object. In case the ray hits opaque voxels, the ray is stopped and the radiance $L(\mathbf{x}, \boldsymbol{\omega}_i)$, is set to zero or to a predefined ambient value to compensate the missing indirect light. Optionally, one or more secondary rays are cast recursively, as known from offline rendering. As the voxelization stores normals and material coefficients, this is supported and used in Figure 7.1(center). Note the reflected wheels on the body of the Toy Car.

7.4.4 *Differential Background Rendering*

For a coherent rendering, we also need to estimate the influence of the virtual object on the environment. Therefore, we use differential rendering to compute a difference image from two global illumination simulations (see Section 4.2). Both simulations require to solve the integral of the Equation (2.12), but at this point, we are missing the material coefficients for the BRDF. Thus, the reflectance parameters for visible surfaces must be estimated before computing the integrals. Therefore, we assume that the environment material is Lambertian. Hence, the BRDF is independent of the observer direction $\boldsymbol{\omega}_o$ and specified by the constant reflectance coefficient ρ_d. Equation (7.5) is then solved for ρ_d [Deb98]. This involves an integral over the hemisphere above a real surface point $\mathbf{x}$ as shown in Figure 7.7b, where $L(\mathbf{x})$ is equal to the color visible in the corrected input image:

$$L(\mathbf{x}) = L(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\mathcal{H}_+} L(\mathbf{x}, \boldsymbol{\omega}_i) \frac{\rho_d}{\pi} \cos\theta_i \, \partial\boldsymbol{\omega}_i \tag{7.5}$$

$$\rho_d = \pi \frac{L(\mathbf{x})}{\int_{\mathcal{H}_+} L(\mathbf{x}, \boldsymbol{\omega}_i) \cos\theta_i \, \partial\boldsymbol{\omega}_i}. \tag{7.6}$$

Now, the influence of the virtual object, $\Delta L(\mathbf{x})$, can be estimated by solving an integral similar to Equation (7.5), where $\Delta L(\mathbf{x}, \boldsymbol{\omega}_i)$ is queried for each incoming direction instead of $L(\mathbf{x}, \boldsymbol{\omega}_i)$. The three different cases that can occur are illustrated in Figure 7.8.

Ray 1 does not hit the virtual object and ends in the environment. In this case, the influence in incoming radiance $\Delta L(\mathbf{x}, \boldsymbol{\omega}_i)$ at the surface point is zero.

Ray 2 hits the virtual object before it hits the environment, $(d_{v_2} < d_{r_2})$. Here we gather the radiance from the environment as well as the radiance of the object and compute $\Delta L(\mathbf{x}, \boldsymbol{\omega}_i) = L_{r+v} - L_r$. Note, that the object radiance, L_{r+v}, is generally not known for rays from arbitrary directions, so we again could stop and return zero or an ambient term. But instead we trace a reflected ray into the environment and compute an indirect light bounce. In this case the object radiance could be referred to as L_v instead. For a correct result, this bounce needs to be computed like in Section 7.4.3. By assuming a diffuse virtual object (only for this bounce), we can achieve color bleeding (e.g., on the wall in Figure 7.10), but we neglect specular effects, like caustics, that appear when bright light is reflected on highly specular objects, such as in

(a) Virtual Object (b) Real-world Environment

Figure 7.7: Normal Estimation Pipeline
Integrating the incoming radiance on a virtual object for rendering (a) and on a real surface for material estimation (b). In both cases, multiple rays are used to sample the upward hemisphere above a position $\mathbf{x}$ on the surface.

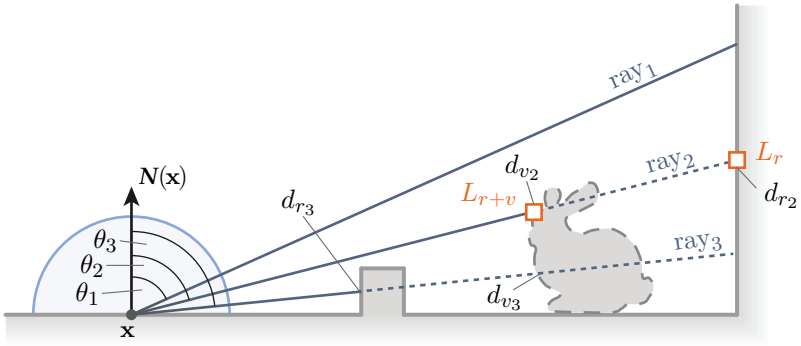


Figure 7.8: Cases for Differential Rendering
Estimating the difference in incoming radiance ΔL for different directions based on the ray distance d. Three cases can be identified.

a metal ring on a table. Hence, the object radiance also contains indirect light from the environment and therefore, the notation L_{r+v} is used in accordance to Section 4.2.

In the last case, represented by ray 3, the ray hits the environment before the virtual object ($d_{v_3} > d_{r_3}$) so $\Delta L(\mathbf{x}, \boldsymbol{\omega}_i)$ is also zero. Note, that these kinds of early occlusions are the main source of artifacts when using impostor tracing. False positive and false negative visibility tests are likely, as the correct ray directions might not be found or not be available.

Depending on the technique and data structure, this ray casting is implemented differently, but the concept is the same in all cases.

7.4.5 *Extension to Cone Casts*

To speed up the ray tests and to reduce the noise, multiple rays can be handled in bundles called cones. Compared to a ray in Figure 7.8, a cone can cover multiple cases at once (also see 2.7.8). Assume ray 1 and ray 2 are the boundaries of such a cone. Then the environment is visible to a degree $\alpha \in [0, 1]$ and the influence of the object is weighted with $1 - \alpha$. With $\overline{L}_{r+v}$ and $\overline{L}_r$ representing the average over the cone, the differential radiance becomes:

$$\Delta L(\mathbf{x}, \boldsymbol{\omega}_i) = (1 - \alpha)(\overline{L}_{r+v} - \overline{L}_r). \tag{7.7}$$

To obtain $\overline{L}_r$ we use the mipmap levels of the environment representations. For VCT and EM we are following the known implementations in [Cra+11, Fra14a] and [CK07] respectively. In case of DIT, the search for the corrected central ray of the cone is performed on the most detailed level of the DI. Only the final texture fetch to gather the radiance is done in a coarser level, which depends on the size of the solid angle and the distance to the surface.

For $\overline{L}_{r+v}$, we start and end cone marching at the boundaries of the bounding box of the object and use step sizes and mipmap layers of the voxelization, depending on the cone radius. Each voxel contains the probability τ for a surface intersection. If the probability τ is greater than zero, the object radiance is estimated using one of the approaches explained in Section 7.4.4 (ray 2 case). The result is weighted with the probability and the cone marching is proceeded. The following steps must be weighted with the probability of continuation α, which is the product $\prod_s (1 - \tau_s)$ of all previous steps. These previous steps can contain the occlusion between cone origin and object (ray 3

case). Finally, when the cone has left the object volume, α is the weight for the environment as given in Equation (7.7).

7.4.6 *Composition*

During composition, the differential image and the virtual object are added to the camera image in reference color space. Then, the inverse color correction c_t^{-1} (see Section 7.2) is applied to the augmented image to transform it back into the input color space.

Figure 7.9 shows results and the capability of the compensation. While the image in the top center shows the augmented original image, we altered the input image by changing the brightness and introducing a strong color shift. Applying the inverse operation to the augmented image also transforms the colors of the virtual object, which results in a coherent integration of the new content.

7.5 ILLUMINATION ESTIMATION USING SAMPLING

The integral of a function $f(x)$ can be estimated by a stochastic process called *Monte Carlo Integration*, as described in Section 2.6.2. Therefore, the integral is approximated by sampling, using Equation (2.21):

$$\int_a^b f(x)\ \partial x \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}, \qquad \text{(2.21 revisited)}$$

where $p(x_i)$ is the probability to produce sample x_i. Any PDF p can be used, but having a function which scales with f yields faster convergence. Using such an improved distribution function is called *Importance Sampling*. For rendering the virtual object, Equation (2.12) is integrated on the GPU, using the BRDF to derive a PDF, as described in [CK07].

For the material estimation, we need to solve Equation (7.6). Since $L(\mathbf{x}, \boldsymbol{\omega}_i)$ is unknown, we use the material-based cosine distribution:

$$p(\boldsymbol{\omega}_i) = {}^1\!/\!_\pi \cos\theta_i.$$

Here, θ_i is the angle between the surface normal of the G-Buffer and the produced direction of $\boldsymbol{\omega}_i$. Applying Monte Carlo integration gives:

$$\rho \approx \pi \frac{L(\mathbf{x})}{\frac{1}{N}\sum_{i=1}^{N}\frac{L(\mathbf{x},\boldsymbol{\omega}_i)\cos\theta_i}{{}^1\!/\!_\pi\cos\theta_i}} = \frac{L(\mathbf{x})}{\frac{1}{N}\sum_{i=1}^{N}L(\mathbf{x},\boldsymbol{\omega}_i)}.$$

Applying the same scheme to Equation (7.5), while sampling the differential values $\Delta L(\mathbf{x}, \boldsymbol{\omega}_i)$ directly (see Section 7.4.4) yields the estimation for the differential rendering:

$$\Delta L(\mathbf{x}, \boldsymbol{\omega}_o) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{\Delta L(\mathbf{x}, \boldsymbol{\omega}_i)\,{}^\rho\!/\!_\pi\cos\theta}{{}^1\!/\!_\pi\cos\theta} = \frac{\rho}{N}\sum_{i=1}^{N}\Delta L(\mathbf{x}, \boldsymbol{\omega}_i)\,. \qquad (7.8)$$

Further, we can improve the sampling by incorporating the knowledge that $\Delta L(\mathbf{x}, \boldsymbol{\omega}_i)$ is zero when the ray misses the object (ray 1 case in Figure 7.8), which is related to observations made in Chapter 6.

Figure 7.9: Transforming Back into Input Color Space
Original input image augmented by a Buddha statue (top center). The other images show artificially altered input images, that have been transferred automatically into the reference color space and eventually, after the augmentation, back into the input space.

7.5.1 *Caching of Samples*

Instead of just skipping rays with no possible contribution, we try to keep the GPU utilization high by tracing other more promising rays. This influences the probability of selecting a single ray, which must be compensated for.

First, we split the set of traced cones into two sets: a set $\mathcal{F}$, containing all cones that miss the object (ray 1 case) and a set $\mathcal{H}$ holding the remaining cones. Then, we divide $\mathcal{H}$ into disjoint sets $\mathcal{H}_a$ and $\mathcal{H}_b$ dependent on ray cases 2 and 3. Accordingly, Equation (7.8) can be split into three parts (omitting $\mathbf{x}$ for compactness):

$$\Delta L(\boldsymbol{\omega}_o) \approx \frac{\rho}{N}\Bigg(\underbrace{\sum_{\boldsymbol{\omega}_i \in \mathcal{H}_a} \Delta L(\boldsymbol{\omega}_i)}_{L_{\mathcal{H}_a}} + \underbrace{\sum_{\boldsymbol{\omega}_i \in \mathcal{H}_b} \Delta L(\boldsymbol{\omega}_i)}_{L_{\mathcal{H}_b}} + \underbrace{\sum_{\boldsymbol{\omega}_i \in \mathcal{F}} \Delta L(\boldsymbol{\omega}_i)}_{L_{\mathcal{F}}=0}\Bigg). \tag{7.9}$$

By using early intersection tests with the bounding box of the virtual object, we can reject all samples in $\mathcal{F}$ before starting expensive tracing. Consequently, the number of samples in $\mathcal{H}$ is decreasing with the distance to the virtual object. To keep the GPU utilized, we increase N and create a large set of N directions, test them against the bounding box and record them – in case of intersection – for later tracing in a local array of size $|\mathcal{H}_a| < N$. In the experiments $|\mathcal{H}_a|$ is usually 16 or 32 and is referred to as the cache size. Now, it is possible that more than $|\mathcal{H}_a|$ valid samples are found for certain pixels. In that case, $|\mathcal{H}_a|$ directions are recorded randomly and the remaining $|\mathcal{H}| - |\mathcal{H}_a|$ are added to set $\mathcal{H}_b$.

$L_{\mathcal{H}_a}$ and $L_{\mathcal{H}_b}$ from Equation (7.9) are estimates of the same integral. Hence, we can discard all samples in $\mathcal{H}_b$, if $L_{\mathcal{H}_a}$ is scaled accordingly. I.e., we assume that on average the samples in $\mathcal{H}_a$ and $\mathcal{H}_b$ fetch the same values. The final equation then becomes:

$$\Delta L(\mathbf{x}, \boldsymbol{\omega}_o) \approx \left(1 + \frac{|\mathcal{H}_b|}{|\mathcal{H}_a|}\right) \frac{\rho}{N} \sum_{\boldsymbol{\omega}_i \in \mathcal{H}_a} \Delta L(\mathbf{x}, \boldsymbol{\omega}_i)\,.$$

Consequently, we maximize the use of a limited number of traced samples with a joint importance sampling of the material (the cosine distribution) and the object. Still, our sampling can produce weak results, if $\Delta L(\mathbf{x}, \boldsymbol{\omega}_i)$ varies a lot. This happens if only a few samples hit very bright regions in the environment map behind the object or if existing bright regions are missed completely.

Table 7.1: Performance of the Acquisition Phase
Timings measured on a Google Tango Developer Kit. As in all examples, the point cloud size is limited to 4M environment samples. Depth information is only provided every 5^{th} frame involving the steps marked by $(\cdot)^*$.

Computation Step	Time in ms
Get Color Image	6.3
Update c_t + Apply c_t	6.0 + 2.5
G-Buffer Splatting + Push-Pull	28.0 + 2.5
Get Sparse Depth Image	4.0*
Find nearest Neighbors	25.8*
Normal Estimation	18.8*
Add Samples to Point Cloud	8.9*
Remove Samples from Point Cloud	5.3*
Apply c_t^{-1}	2.5
Others (Visualization, ...)	13.8
(Averaged) Total Frame	(73.4) 123.6*

7.6 Results

Here, we discuss the results of the different stages in the order of the previous sections. To evaluate the presented approach, we investigated all stages of the pipeline in real-world scenarios and in synthetic scenes, acquired by a simulated depth sensing device. Hence, even the synthetic experiments perform all steps of the pipeline. To provide insights on the performance of the system, we show results created on a desktop PC with an Intel i7-4790S and an Nvidia GeForce GTX 980. The mobile device, that we are using is a Google Tango Developer Kit powered by an Nvidia Tegra K1.

7.6.1 *Environment Acquisition*

As the synthetic scenes showed no issues, we are focusing only on real data. Therefore, we mainly refer to the accompanying video in Appendix A.3. Visual results can also be found in Figures 7.2 and 7.3. The depth sensor provides between $3,000$ and $14,000$ new depth samples every 5^{th} frame in which the complete acquisition pipeline is executed. During the other frames, we also provide feedback to the user by showing the point cloud from the current camera position. The timings provided in Table 7.1 show that the average frame time, including frames with and without depth sensing, is 73.4 ms which equals about 13.5 Hz.

While larger errors in the position of the samples are mainly caused by the given tracking, we concentrate on issues with the measured radiance. Using our compensation estimation, we measure the radiance in the scene relative to the first frame. The compensation provides a stable color adjustment in consecutive frames but, as known from related algorithms like SLAM, there is a drift caused by error accumulation over time. This can be observed when closing the loop of 360°. We observed increasing errors in scenes where the camera gets over-saturated by bright light sources or under-saturated in very dark areas causing noisy images. Another issue we identified is vignetting, a radial reduction of brightness in the input. Our compensation is not able to deal with these non-local effects.

Table 7.2: Performance of the Rendering Phase
Timings are measured in ms for the Toy Car in Office scene. All three quality settings are shown in Figure 7.1. The *med* quality setting is also used in the video of Appendix A.3.

Platform	Tango	Tango	Tango	PC	PC	PC
Technique	EM	EM	DIT	DIT	VCT	DIT
Quality Settings	*low*	*med*	*med*	*med*	*med*	*high*
Computation Step						**Time in ms**
Get Color Image	5.8	7.2	6.2	–	–	–
Update c_t	7.2	10.3	7.8	1.1	1.1	1.2
Apply c_t	3.0	2.5	2.7	0.1	0.1	0.1
G-Buffer Splatting	34.7	30.5	30.5	8.1	8.1	8.3
G-Buffer Push-Pull	2.6	1.8	1.8	0.4	0.4	0.4
Render Virtual Object	18.7	149.6	195.0	5.1	8.6	133.2
Material Estimation	45.7	45.5	45.4	1.1	1.2	18.6
Differential Background	22.5	32.5	61.7	0.8	2.0	16.9
Apply c_t^{-1}	3.0	2.5	2.7	0.1	0.1	0.1
Others (Compose, ...)	4.4	4.5	4.1	0.2	0.4	0.5
Total Frame	147.6	288.5	357.9	16.9	21.9	179.0
Frame Rate [Hz]	6.7	3.4	2.8	59.1	45.6	5.6

However, the strongest discrepancies are due to the Lambertian material assumption, as real surfaces usually show reflections and highlights. As the effects did not appear in the synthetic experiments, we can assume that there is no systematic error even though we are not able to provide a real-world ground truth comparison.

7.6.2 *G-Buffer Generation*

Splatting the entire point cloud every frame is one of the bottlenecks that is not addressed so far. Besides the computation time (see Tables 7.1 and 7.2), there are also qualitative issues. Splatting densely sampled areas can lead to z-fighting, which is acceptable for colors, but results in temporal incoherence in the depth and normal channels. Applying bilateral Gaussian filters or down-sampling solved the problem in our experiments. We use 320×180 as G-Buffer resolution in all experiments. Sparsely sampled areas, e.g., because of glossy surfaces, are smoothly filled by the push-pull steps. Large gaps however, especially at the viewport boundaries can lead to serious artifacts. Even in moderate cases the surface is bumpy. Figure 7.1(center) shows bright spots close to the papers that are not shadowed. This is an important point for further research. Note, that the proposed normal estimation could also improve the G-Buffer but at a high cost.

7.6.3 *Scene Representation*

Splatting the point cloud into a distance impostor follows the generation of the G-Buffer. The timings are similar and scale linearly with the number of pixels. However, DIs are only updated when the virtual object is moved and computations can be distributed over multiple frames. Hence, temporal

coherence can only become a problem, if the object is moved. If not mentioned otherwise, we use a resolution of 6×256^2 for DI or (scene) voxel volumes of size 256^3.

7.6.4 *Rendering of the Virtual Object*

For performance evaluation, we refer to Table 7.2. The scene with the TOY CAR, visible in the video and Figure 7.1, is rendered using different quality settings. *Low* in Figure 7.1(left) and *med* in Figure 7.1(right) use 16 samples for material estimation, differential rendering (with cache size 8) and rendering of the virtual object. In *high* settings of Figure 7.1(center), the sample counts are increased to 64 for material estimation as well as differential rendering (with cache size 32) and to 128 for rendering the virtual object. The TOY

(a) Comparison of EM, DIT and VCT (b) Ground Truth

Figure 7.10: Illumination Variants for the Virtual Object
The rows of (a) contain results without and with self-occlusion tests for the different techniques: Environment Mapping (top), Impostor Tracing (middle) and Voxel Cone Tracing (bottom). The ground truth solution (b) for comparison is created using an offline path tracer.

Table 7.3: Environment Query Timings
Accumulated time in ms for rendering the virtual object, material estimation, and computing the differential background (with cache size 32) on PC (also see Figure 7.10).

Sample Count N		16	32	64
Computation Step			Time in ms	
EM	(no visibility test)	0.27	0.48	0.99
EM		9.88	20.49	42.12
DIT	(no visibility test)	3.57	6.91	13.82
DIT		12.81	25.81	51.86
VCT	(no visibility test)	6.43	12.14	27.01
VCT		18.97	37.72	79.24

Car consists of 230k triangles and is voxelized into a volume of size 32^3 for *low* and *med* but $128 \times 96 \times 96$ for *high*. The *low*-quality setting additionally omits self-shadow tests. With *med* settings, we are checking for visibility only and for *high* we trace a secondary bounce. Compared to visibility test only, the secondary bounce takes approximately twice as long for rendering the virtual object. Note, that for the material estimation DIT is also used in EM cases, as EM cannot be applied here. All measured timings are reasonable but show that the mobile device is barely able to reach interactive frame rates even with low quality settings.

Figure 7.10 shows the Buddha model in a synthetic test scene, which is used for qualitative comparison of the alternative techniques. We evaluate a diffuse material as they reflect light from all directions and tend to show more noise in the sampling. The results show, that self-shadows appear significantly more plausible in all three approaches, but also increase timings notably (see Table 7.3). Secondly, the simple EM technique, which assumes distant light, illuminates the object uniformly from top to bottom. In both other techniques, the origin of the ray or cone is considered, which leads to a more correct variation in brightness. Table 7.3 contains the timings for different settings, where gray cells correspond to the images in Figure 7.10.

7.6.5 *Material Estimation*

We assumed a Lambertian environment, which allows to integrate the incident light over the upper hemisphere of the surface. Since this assumption is not valid in real world scenarios we are not able to achieve perfect results. Another problem is caused by an incomplete acquisition of the environment. Windows or bright sources of light cannot be measured by the Tango device. Therefore, we added a workaround to add guessed environment samples. Triggered by a manual action of the user (see the accompanying video in Appendix A.3), the current camera image is projected along the view direction and samples are created at the distance, estimated by the hole-filled depth buffer. These samples are excluded from the compensation estimation but are used for rendering and material estimation. Since these samples are often created in saturated areas of the camera image, the real – often much brighter – radiance is underestimated. Hence, the material coefficients will be overestimated and shadows computed by differential rendering appear different.

Figure 7.11 shows the entire pipeline, including the estimated material parameters as well as the resulting difference image. Besides the color shift on the right side of the image, the material still contains some shadows but removed the shading to a large extent. Ideally, the material image does not show any lighting or shading and rather look like textured by albedo maps.

7.6.6 *Differential Background Rendering*

To evaluate the relighting of the background, we measured different timings with varying parameters in the synthetic scene (see Table 7.4 and Figure 7.12). The images show an equal time comparison of EM, DIT and VCT. The resolution for the differential image is reduced by a factor 2 to 640×360. One bounce of indirect light is computed in case the virtual object was hit.

Obviously, none of the techniques can reproduce the reference solution. VCT and DIT at least provide contact shadows, whereas EM yields a uniform direc-

(a) Camera Input

(b) Reference Color Space

(c) Reflectance Coefficients

(d) Differential Image (absolute values)

(e) Augmented Image in Reference Space

(f) Augmented Image

Figure 7.11: Compensated Differential Rendering
The camera captures input images in an unknown input color space (a). These images are transformed in HDR reference space (b) and serve as input for the estimation of the reflectance (c), which can contain errors because of saturation in the input image. The differential image (d), containing the influence of the virtual object (absolute value), is applied to the augmented image in HDR reference space (e). Eventually, the result is transformed back into the unknown input color space (f) Model courtesy of Van Quang Ho.

Table 7.4: Computing Virtual Influence

Timings in ms for different number of samples and cache sizes to estimate the differential rendering integral, using EM, DIT and VCT on PC (also see Figure 7.12).

Sample Count N		8	16	32	64	128
Computation Step					Time in ms	
EM	(Cache 8)	2.15	2.62	3.33	3.85	4.46
EM	(Cache 16)	2.16	3.10	4.90	6.44	7.36
EM	(Cache 32)	2.19	3.11	5.88	9.51	12.48
DIT	(Cache 8)	2.25	3.27	4.16	4.67	5.51
DIT	(Cache 16)	2.74	3.86	6.05	7.87	9.86
DIT	(Cache 32)	2.76	3.85	7.14	11.56	15.50
VCT	(Cache 8)	8.37	3.98	5.10	5.73	6.49
VCT	(Cache 16)	8.34	4.71	7.56	9.92	11.10
VCT	(Cache 32)	8.45	4.71	8.77	14.57	19.23

tional shadow because of the distant light assumption. Even though this is incorrect, the resulting shadow is smooth and can be used as a fast approximation. Increasing the number of cones and reducing their radius, generally improves the visual appearance.

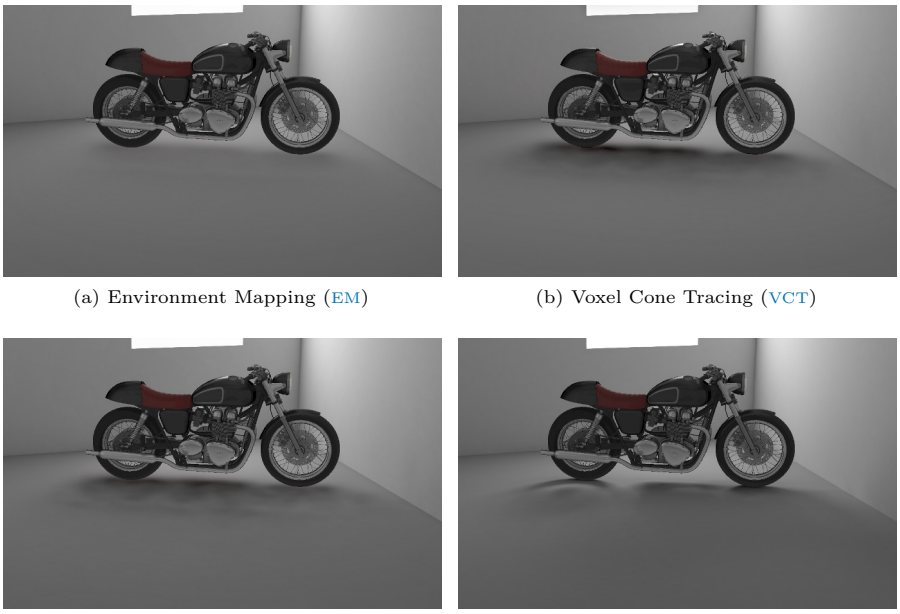
Because of the diffuse assumption and the low number of cones, the probability to hit a light source is low and the cone radius must be large to properly sample the entire hemisphere. Hence, we are not able to reproduce high frequency shadow details. Compared to the ground truth, the direction and the coarse shape are correct though. In other rendering engines, VCT is used for indirect light only. Therefore, five to eight samples are often considered as sufficient. We also handle the direct illumination. In the future, hybrid solutions with extracted light sources can be desirable. However, extracting them from the point cloud is not straightforward.

Table 7.4 also shows the impact of the presented caching approach, that allows to select sample directions, not only based on the BRDF but also on the direction of the virtual object (see Section 7.5). The measurements confirm an increase of the utilization and thereby a reduction of cost per sample. The costs increase as long as the cache contains less than $|\mathcal{H}_a|$ samples (see Equation (7.9)). Afterwards, the costs per sample equal the cost of the box test. For temporal coherence, we refer to the accompanying video (see Appendix A.3), showing the moving Buddha and DIT with $N = 64$ and a cache size of 32.

Interreflections between the real world and inserted virtual objects are another important feature of coherent AR rendering. By computing a secondary bounce with any of the three techniques, we are able to produce color-bleeding, e.g., on the wall in Figure 7.10.

7.6.7 *Composition*

For visual results and an impression of the visual behavior we again refer the reader to the supplemental material (Appendix A.3). During the acquisition of the environment the colors are matched frame by frame. In the sequence containing the Toy Car, a visually plausible, automatic adaptation to the camera exposure can be observed. An image of that sequence is also shown in



(a) Environment Mapping (EM)

(b) Voxel Cone Tracing (VCT)

(c) Distance Impostor Tracing (DIT)

(d) Ground Truth

Figure 7.12: Differential Shadows
Shadow quality achieved within 10 ms of differential background computation: EM (a), VCT (b), DIT (c), and the ground truth (d) computed using an offline path tracer.

Figure 7.1(bottom right). Figure 7.9 shows that the approach is also robust against different color transformations on the input. Therefore, our compensation is of great use in many AR scenarios.

However, if the input is already saturated, problems will occur. In this case the transformation leads to undefined colors in the corrected image, but the material estimation relies on them. Figure 7.11 shows the effect on the resulting reflectance coefficients. Since the resulting values are stored in HDR, this is not visible to the user, as long as these areas do not receive virtual shadow.

Timings for updating c_t can vary strongly (see Table 7.1 and 7.2). This is because the timing contains a read-back from the GPU to disable acquisition in case of failures (shared code for rendering phase). Setting up and solving the system on the GPU itself requires about 2.6 ms on the Tango and only depends on the G-Buffer resolution.

7.7 Discussion

To be able to demonstrate a coherent AR rendering starting from the acquisition using a single mobile device, we need to make a set of assumptions that are not necessarily true in real-world scenarios.

The compensation estimation relies on a linear model to map input images into a given reference color space. In our experiments, this approach was sufficiently robust, even in difficult situations, e.g., when turning the camera directly to the light. However, for other devices or different applications, a linear compensation might not be adequate, e.g., if the transformation is not constant over the entire image. Further, we are limited by the dynamic range of the Tango. When pointed to a bright light source, the exposure is reduced to

Figure 7.13: Another Result with High Quality Settings
The Cafe Racer model with mirroring chrome elements that reflect the environment.

not over-saturate. When this turns the rest of the image too dark or even black, the matching must fail. Recently, Eilertsen *et al.* presented a CNN-based approach to recover HDR information from saturated LDR images [Eil+17]. Even though, this network can only guess the radiance of light sources, such an approach can be used as fall-back to deliver at least approximations, when other methods fail. In any case, this technique can be applied to our approach during the material estimation, which also suffers from over-saturated input (see Section 7.4.6). Here, even guessed HDR values, can only improve the resulting estimated coefficients.

The most restrictive assumption is the Lambertian environment BRDF. This affects the acquisition process, where the captured environment is used as input to the color estimation in consecutive steps. Reflections on non-diffuse surfaces introduce errors, that cannot be handled by the system at the moment. Also, the material estimation required for differential rendering depends on this assumption. Including additional information about the direction of measurement can help to overcome this problem by measuring more complex material properties. Further, the sampling quality will benefit from smaller cones due to more focused reflections.

Although we are able to remove samples from the point cloud, the environment is assumed to be static after completing the acquisition. During the augmentation, the acquisition pipeline could update the environment model, but this only works for the small field of view of the camera. Changes anywhere else stay unnoticed until seen for the first time. By simply turning a light source on or off, the entire reconstruction becomes outdated. Measuring the illumination with additional hardware (see Chapter 5) or inverse rendering techniques can be used here.

As noticed earlier, the reconstructed surface can be very bumpy. This can be addressed by taking the previous acquired scene into account, while processing new samples, e.g., by applying ICP. Up to now, the normal estimation works directly on new incoming measurements without considering the samples that have been captured in the past. By also incorporating the information of previous frames or the already captured surfaces, the geometry reconstruction

can be improved. A simple temporal filtering can already lead to significantly smoother data in the G-Buffer. While accepting an additional processing step, a surface reconstruction [KH13], will allow for mesh processing operations, that result in a smoother reconstruction. In this context, mesh simplification can also be applied to improve the performance, while filling the G-Buffer and the distance impostors.

The presented rendering techniques allows for different levels of quality. However, the tablet used in the experiments achieves barely interactive frame rates, even with low settings. There is a lot of potential in optimizing the performance, as we aimed for flexibility instead of polishing a single technique. However, different bottlenecks can be addressed to achieve higher frame rates. Processing and optimizing the acquired scene, e.g., by reducing samples in dense regions, creating hierarchical structures, that allow culling and fast traversal, estimating reflectance coefficients offline or creating a low poly mesh.

At this point, only one virtual object can be inserted into the image. Adding multiple objects with multiple voxel volumes also multiplies the computation effort. Using a hierarchy of boxes that supports importance sampling is a challenging task for future research.

We apply several sampling-based techniques for high quality renderings based on the acquired data. While aiming for interactive augmentations on mobile devices, we presented a framework that scales with the performance of the available platform up to high quality renderings, as shown in Figure 7.1 and 7.13. Note, that progressive rendering to produce high quality screenshots, e.g., for sharing, is a straight forward extension. Multiple augmented images are computed with different random seeds. Averaging the individual results leads to a progressively refined solution. By also moving the virtual camera within the range of half a pixel, anti-aliasing is added without further computation effort. Figure 7.14 shows a comparison of a real-world photograph to such a progressively refined augmentation, computed in 20 s.

It is possible to use natural environment illumination with different devices involved. As long as the camera pose and the intrinsic camera parameters are known, images of any other source can be transformed into the reference color space and back. So, the color compensation is not limited to be used

(a) Progressive Rendering (b) Real-world Photograph

Figure 7.14: Progessive Rendering
Comparison of a progressive rendering (a) and a photograph (b) of a porcelain figurine. The model was acquired using a 3D scanning device and the reflectance was manually specified. Note, that the photograph was taken on a day with overcast sky, while the scene reconstruction for rendering was already made on a sunnier day.

with depth sensing devices. As an example, the background of Figure 7.1 was captured with a DSRL camera.

Thinking of a high-quality scan of an area of interest – maybe a historic sight or a room of a museum – a common smartphone can display augmentations based on that scan. Therefore, extrinsic camera parameters can be estimated from visual features and the presented estimation method is used to map the camera image into the color space of the high-quality scan. After applying the augmentations, the inverse compensation is used to transform back into the unknown color space, that was optimized for displaying the real content.

We consider an offline processing like presented by Zhang *et al.* [ZCC16] an optional extension to the presented method. Compared to our frame-to-frame exposure compensation, an offline step can consider all input frames, which leads to a global solution and coherent compensation with closed loops. Naturally, this improves the quality of the measured radiance for all environment samples, which also improves the estimated reflectance and potentially the display quality of reflections on virtual objects.

8

CONCLUSION

In this thesis, I presented a set of methods embedded in two frameworks for visual coherent AR rendering. By capturing the real-world light condition of the local environment and making the acquired information available to real-time rendering pipelines, the methods lead to an improvement in visual quality for augmented and mixed reality applications. Mobile devices with their increasing computing capabilities and a variety of available sensors, are currently one of the most important target platforms for AR. Hence, they were the primary focus during the development of the presented methods. However, to ensure the long term relevance of this work, scalability was also one of the major design criteria. The techniques always offer a trade-off between performance, quality and resource consumption. This allows future hardware generations to apply the same techniques at higher frame rates, lower energy consumption or increased visual quality. The frameworks were also developed with the claim to be applicable in a broader context. Individual steps can be computed in advance and consider pre-computation to avoid complex hardware setups (see Section 5.7). This adds additional restriction in terms of support for dynamic environments, but allows to expand the field of application significantly. Another example is the support for individual and specialized devices, used to acquire the scene information (see Section 7.7). While an all-in-one solution aims towards day-to-day application scenarios, a separation of acquisition and presentation can provide higher quality scene reconstructions, used for augmentations on consumer hardware that is already available to the user and thereby allows a wide field of applications today.

8.1 SUMMARY

The goal and the expected advantages of a visually coherent presentation in AR applications were motivated in Chapter 1. By showing the differences to the currently very popular field of virtual reality and also to the composition of virtual and real-world footage in the area of special effects for movie productions, this first chapter also defines the term AR in accordance to AZUMA based on three criteria: the combination of real and virtual, interactive in real-time and registered in 3D [Azu97]. The motivation is continued by several example applications for AR in general and for AR with photorealistic, coherent visuals, while this thesis is highly relevant for various applications in this last group – including industrial design, architecture, furnishing, education, retail, entertainment and cultural heritage. The chapter also includes a summary of contributions, a list of publications and a brief overview of the rest of the thesis.

The most important aspect of coherent rendering is a physically-based light transport simulation. Therefore, Chapter 2 is used to convey the basic foundation of photorealistic rendering, starting with radiometric quantities, and the concepts of geometric optics, that allow to compute the transport of light in the scene. This is followed by KAJIYA's rendering equation [Kaj86] and BSDFs to model the interaction between light and matter. The second part of this chapter provides an overview on light transport simulation techniques, that make use of this theoretical framework to eventually create computer generated images, which in the best case scenario, cannot be distinguished from photographs. The overview includes classical approaches, that are primarily used to create correct solutions, and real-time rendering techniques, which approximate the light transport to allow for interactive applications, for instance in the context of AR.

Chapter 3 forms the second part of the foundation of this thesis, focusing directly on AR and the requirements of this interdisciplinary field. Based on the book of SCHMALSTIEG and HÖLLERER [SH16], this chapter covers the three aspects of visual coherence: geometric registration, photometric registration and camera simulation. Since AR requires the use of cameras in order to provide a composition of real and virtual elements and to be able to reconstruct and to reason about the environment itself, these cameras need to be calibrated to align with a selected camera model. Therefore, the camera models used in this thesis, as well as calibration techniques to estimate intrinsic, extrinsic and radiometric parameters are discussed in a detail that allows to apply the individual models to the presented AR techniques. The chapter concludes with an overview of different types of AR displays, their characteristics and an assessment of the relevance of this thesis with regard to the different types.

The related work of this dissertation is mainly discussed in Chapter 4. As a reliable reconstruction of the geometry, materials and lights of the environment is the key requirement for coherent rendering, various approaches, starting with the abstract and idealistic *Plenoptic Function*, are discussed to provide an overview of currently available methods to create such a reconstruction. Based on that, the concept of *Differential Rendering* as described by FOURNIER *et al.* and DEBEVEC is introduced as the basic principle to create augmented images by applying light transport simulations [FGR93, Deb98]. Important or unique interactive approaches, as well as mobile approaches, presented in recent years, have been selected and described, to provide an

overview of the related work in interactive coherent AR rendering, and to classify the own approaches in the context of the current state of the art.

The framework for *Distributed Near-Field Illumination* is the topic of Chapter 5 and represents the first part of the contribution of this dissertation. Motivated by limited computational capabilities of mobile devices compared to desktop hardware, and the physical limitations on the area that can be observed using a single device, I suggested a distributed approach, to share the required computations among participating devices. More precisely, a stationary machine equipped with multiple cameras to continuously capture the radiance on real-world surfaces is used to acquire the dynamic environment. The gathered information is processed to extract parameters for the lightweight illumination model, that are then transferred to one or multiple mobile devices for interactive AR rendering. Therefore, the environment light is split into two parts: First, a set of global area light sources, which represent the high-frequency direct and strong indirect illumination. These sources cast shadows from real onto virtual objects and vice versa. And second, the remaining low-frequency illumination, which is compressed into SH basis for each virtual object and PRT-based illumination. Both parts can be evaluated on the mobile devices and provide a consistent appearance of virtual objects – including *color bleeding* from real to virtual surfaces – at interactive speed and without delays in a possible streaming-based solution. Since all illumination parameters are broadcasted via WiFi, there is no overhead for additional clients that display individual views into the augmented scene. Furthermore, the number of extracted direct light sources controls the visual quality, which is particularly noticeable in the area of shadow edges, as soft shadows are created by overlapping hard shadows.

Chapter 6 deals with the culling of light sources. The key aspect for optimizing the performance of the system, is reducing the number of sources, that have to be considered during the computation of the differential illumination. This, in turn, improves visual quality without introducing bias while achieving the same or even higher frame rates. To realize this culling strategy, a data structure was introduced, that allows to explicitly store the relevant light sources in 3D spatial data structure. Applying the proposed strategy achieved a performance gain of factor 2.2 and encouraged the use of the more recently proposed *Tiled Forward Rendering* [OA11], which in turn supports alpha blending used for transparent objects and multisampling for anti-aliasing. The proposed culling can also be applied to other point light based AR techniques like *Differential Instant Radiosity* [Kel97]. In the context of VR and interactive videos, the same strategy can be applied to render interactive content into virtual scenes with baked lighting and real-world video footage for instance.

The *Natural Environment Illumination* framework is the focus of Chapter 7. It addresses the problem of coherent illumination of virtual objects in the local real-world environment for more practical day-to-day application scenarios, without a complex hardware setup. The estimation of a linear color compensation that models the unknown color transformations, applied by mobile cameras and their drivers, is the foundation of this second framework. It allows to fuse LDR input images, given in an unknown color space, into HDR radiance images in a reference color space that is used for the entire augmentation pipeline. In combination with the depth sensing capabilities of the used mobile device, a spatial light-based scene representation can be recorded, which stores 3D position, fitted surface normals and visible radiance

in point cloud structure. This HDR point cloud is then transformed into specialized data structures that allow for real-time Monte Carlo rendering based on either *Environment Mapping*, *Impostor Tracing* or *Voxel Cone Tracing*. All three strategies have been applied to the estimation of the real-world diffuse reflectance, to illuminate the virtual objects and to compute the differential influence of the virtual objects on real surfaces – in each case using Monte Carlo sampling. This work successfully showed that the concept, which is common practice in offline rendering, can be applied to interactive AR – not least because of an improved sampling scheme. The number of samples used in the individual steps, provides a natural trade-off between performance and quality. The chosen data structure that is used to represent the environment is another degree of freedom in that trade-off. However, since classical environment maps assume distance light and thereby introduce noticeable artifacts in the near-field illumination and due to the memory consumption associated with VCT, distance impostors are the recommended default choice. Nevertheless, the system can be used for interactive AR on mobile devices and scales up to progressive offline rendering to produce high quality augmentations in an offline rendering context. Inverting the previously estimated color transformation and applying it to the augmented image results in a reasonable tone mapping. Most importantly, it yields a seamless blend of virtual and real objects that adapt to the camera parameters, automatically chosen by the driver to deliver pleasant pictures without requiring the user to have solid understanding of photography. Using specialized hardware for the acquisition and consumer hardware for the presentation phase, further expands the range of possible applications of this approach.

8.2 FUTURE WORK

This dissertation showed that coherent interactive AR rendering is possible and able to deliver convincing results, even on mobile hardware. A number of approaches is contributed to the set of tools for coherent rendering in augmented, mixed and even virtual reality. Nevertheless, real-time AR is still a field of largely unexplored areas [Fra15]. Thus, there are many possible paths to follow in the future. Potential extensions of the individual approaches have been discussed at the end of the Chapters 5 to 7, but there are new questions and ideas that arise when considering the combination of the presented methods. Finally, there are more general issues that need to be addressed in future research. The following paragraphs may provide starting points for that discussion:

COMBINED APPROACHES One of the particular properties of the distributed illumination approach is the support of dynamic real-world geometry and lighting, as multiple cameras are able to recognize changes in the environment that are not visible to the client device. The reconstruction, created by the natural environment illumination technique on the other hand, is fixed when entering the presentation phase – assuming both phases are not running in parallel, which is possible in straightforward manner given the required computational power. Combinations of both techniques can be realized by rendering distance impostors on the stationary machine and transferring incremental changes to the clients, e.g., by using a four channel, 360°, HDR video stream. A Monto Carlo-based rendering can then be used for rendering. The latency caused by streaming will in this case only affect the lighting and

not interactive manipulations by the user. Another possible combination of the techniques would be to apply the server pipeline of the distributed approach as a post-process after the acquisition phase in the second framework, maybe even in the context of specialized hardware. This allows to apply the forward rendering, presented in Chapter 6, based on the acquired point cloud data. Replacing the manual geometry and material reconstruction required in the first framework with the point cloud approach of the second one will also expand the applicability of the distributed system. Finally, the color compensation can be used during the rendering in the distributed framework but also in various other existing and future methods.

CAMERA SIMULATION While the presented color compensation technique fits captured data to certain global models, e.g., linear transformations, to anticipate unknown color adjustments, a solution is provided only for one part of the problem. We also need to deal with local effects like vignetting or other global phenomena, such as blurriness, noise or chromatic aberrations. The latter and also lens distortions, Bayer masking and color space compression have been discussed and addressed before by KLEIN and MURRAY [KM08]. Depending on the rendering technique, effects like motion blur and depth of field are not straightforward to realize or require a serious additional effort. Analyzing and reproducing the behavior of an individual camera in motion and depth of field rendering for AR have also been discussed in the literature already [OKY06, KK12b]. However, especially when considering optical see-through devices, these topics will be of particular interest again. Creating standardized approaches to deal with these camera effects will be necessary to achieve the far goal of instant coherent AR at any place.

DISPLAY DEVICES As discussed in Section 3.5.3, this work focused on video see-through devices, for which standard image processing techniques can be applied to all components of the augmented images, including the background camera stream. For optical see-though and spatial AR, there are additional challenges, since it is hard to selectively block light for OSTs and it is not possible to subtract light from the real scene using a projector system. Additionally, view-dependent aspects of the illumination can become problematic in case of multi-user SAR. Similar to the VR case, there are opportunities to exploit redundancies in stereo rendering for binocular head-mounted devices.

GLOBAL ILLUMINATION Many approaches of none-AR real-time rendering, especially the ones developed for games, can be adapted for the use in augmented reality. In contrast to games, where a precise model of the environment is given, we need to deal with dynamic real-world surroundings. It is not obvious if real objects are static structures, if they can be moved in rigid manner or if they allow free deformations. Lights, that have been switched off, may not be recognized as a possible source of light or maybe they have not been visible at all due to occlusions. In games, a deep knowledge about the scene allows for optimization and pre-computations in order to maximize performance and quality, whereas in AR we need to be able to respond to unpredictable changes in the environment and their global impact on the augmented scene. Therefore, we cannot directly apply any real-time technique to AR rendering. However, many of these techniques have been successfully ported, e.g., Instant Radiosity [Kne+10], Voxel Cone Tracing [Fra14a] or the

tiled rendering in our distributed illumination framework. Other techniques such as Micro Rendering [Rit+09, Hol+11] could be interesting to investigate, for instance with the acquired HDR point clouds (see Chapter 7) used as input.

MATERIAL RECONSTRUCTION The developed techniques and most of the approaches discussed in Chapter 4 assume that real-world surfaces show a Lambertian reflectance behavior. This allows to measure a view-independent surface radiance from one observation and the estimation of the reflectance coefficients at that surface element by evaluating one integral over the upward hemisphere. Unfortunately, this rather easy to handle model is technically not valid for any real-world surface. While most materials, especially the man-made, are glossy or specular, even extremely rough and diffuse looking materials show view-dependent reflections when observed from grazing angles. To acquire such a reflectance behavior requires observations from many directions and knowledge about the light condition. A possible step towards online material reconstruction for non-diffuse surfaces could involve the collection of statistics for different viewing angles for each of the observed surface points or clusters that represent material groups. The same applies to the process of capturing a goniometric diagram, i.e., the intensity distribution of a complex light source. However, to deduce parameters of more complex BSDFs, knowledge about the intensity of the reflected light is required, which in turn can also be highly view-dependent. When considering translucent real-world surfaces the problem becomes even more challenging, which offers many opportunities for future research. While it is possible to measure or estimate the material parameters of single objects today, we are far from adequately capturing the parameters for all surfaces in the environment of a typically AR scenario.

The research projects, that are summarized in this thesis, resulted in novel approaches, that enrich the mobile platform by a set of new tools for coherent AR rendering. However, there are many unanswered questions or unresolved issues and thereby various opportunities for future development and research to ultimately fulfill the vision of seamless and coherent interactive AR for use in both, personal and professional applications.

A

APPENDIX

The accompanying DVD contains the supplemental material of the corresponding publications. The supplemental material may also be accessed using the additionally provided URLs.

A.1 DISTRIBUTED NEAR-FIELD ILLUMINATION

DESCRIPTION

Visual results of the technique presented in Chapter 5 in different settings including animated virtual objects, interactive manipulations by the user and results under spatially varying illumination and in the context of moving real-world objects. The material also contains simple description on how the method works and visual comparison with results of the instant radiosity technique.

FILENAME/URL

DistributedNearFieldIllumination.mp4
https://kairohmer.azurewebsites.net/Publications/Details/ismar2014

A.2 TILED FRUSTUM CULLING FOR DIFFERENTIAL RENDERING

DESCRIPTION

The video sequences show visual results of the technique presented in Chapter 6 and different scenarios, including interactive manipulations by the user. Additionally, the animation used for evaluating the presented culling techniques is shown in full length.

FILENAME/URL

TiledFrustumCulling.avi
https://kairohmer.azurewebsites.net/Publications/Details/ismar2015

A.3 NATURAL ENVIRONMENT ILLUMINATION

DESCRIPTION

A sequence that shows the entire acquisition of an office and the results of another scan in a living room. Both are used for rendering several images visible in Chapter 7. The material also contains several other videos of interactive sessions with coherent AR rendering.

FILENAME/URL

NaturalEnvironmentIllumination.avi
https://kairohmer.azurewebsites.net/Publications/Details/ismar2017

BIBLIOGRAPHY

[AB91] E. H. Adelson and J. R. Bergen. “The Plenoptic Function and the Elements of Early Vision.” In: *Computational Models of Visual Processing*. Massachusetts Institute of Technology, 1991, pp. 3–20.

[Aga+03] S. Agarwal, R. Ramamoorthi, S. Belongie, and H. W. Jensen. “Structured Importance Sampling of Environment Maps.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 22.3 (2003), pp. 605–612.

[Agu+03] K. Agusanto, L. Li, Z. Chuangui, and N. W. Sing. “Photorealistic Rendering for Augmented Reality Using Environment Illumination.” In: *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. 2003, pp. 208–216.

[AHB87] K. S. Arun, T. S. Huang, and S. D. Blostein. “Least-Squares Fitting of Two 3-D Point Sets.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 9.5 (1987), pp. 698–700.

[AHH08] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering*. 3rd. A K Peters, Ltd., 2008.

[Ait10] M. Aittala. “Inverse Lighting and Photorealistic Rendering for Augmented Reality.” In: *The Visual Computer* 26.6-8 (2010), pp. 669–678.

[And09] J. Andersson. “Beyond Programmable Shading: Parallel Graphics in Frostbite - Current & Future.” In: *ACM SIGGRAPH Courses*. 2009, 7:1–7:40.

[Ann+04] T. Annen, J. Kautz, F. Durand, and H.-P. Seidel. “Spherical Harmonic Gradients for Mid-Range Illumination.” In: *Proc. Eurographics Symposium on Rendering (EGSR)*. 2004, pp. 331–336.

[Ash93] I. Ashdown. “Near-Field Photometry: A New Approach.” In: *Journal of the Illuminating Engineering Society* 22.1 (1993), pp. 163–180.

[Azu97] R. Azuma. “A Survey of Augmented Reality.” In: *Presence: Teleoperators and Virtual Environments* 6.4 (1997), pp. 355–385.

[Ban+09] F. Banterle, K. Debattista, A. Artusi, S. Pattanaik, K. Myszkowski, P. Ledda, and A. Chalmers. “High Dynamic Range Imaging and Low Dynamic Range Expansion for Generating HDR Content.” In: *Computer Graphics Forum (CGF)*. Vol. 28. 8. 2009, pp. 2343–2367.

[Bät15] F. Bäthge. “Real-time Screen-Space Ambient Occlusion for Mobile Devices.” Master’s thesis. Otto-von-Guericke-Universität Magdeburg, Germany, 2015.

[Bay+08] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. “Speeded-up robust features (SURF).” In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359.

[BE08] C. Balestra and P.-K. Engstad. “The Technology of Uncharted: Drake’s Fortune.” In: *Proc. Game Developers Conference (GDC)*. 2008, pp. 1–58.

[BG01] S. Boivin and A. Gagalowicz. “Image-based Rendering of Diffuse, Specular and Glossy Surfaces from a Single Image.” In: *Proceedings of SIGGRAPH ’01*. ACM, 2001, pp. 107–116.

[BI97] A. Blake and M. Isard. “The CONDENSATION Algorithm - Conditional Density Propagation and Applications to Visual Tracking.” In: *Proc. Advances in Neural Information Processing Systems (NIPS)*. Ed. by M. C. Mozer, M. I. Jordan, and T. Petsche. 1997, pp. 361–367.

[Bie+94] E. A. Bier, M. C. Stone, K. Pier, K. Fishkin, T. Baudel, M. Conway, W. Buxton, and T. DeRose. “Toolglass and Magic Lenses: The See-through Interface.” In: *Proc. Conference on Human Factors in Computing Systems (CHI)*. 1994, pp. 445–446.

[Bli88] J. Blinn. “Me and My (Fake) Shadow.” In: *IEEE Computer Graphics and Applications (CGA)* 8.1 (1988), pp. 82–86.

[Blo12] C. Bloch. *The HDRI Handbook 2.0*. Rocky Nook, 2012.

[BM04] S. Baker and I. Matthews. “Lucas-Kanade 20 Years On: A Unifying Framework.” In: *International Journal of Computer Vision* 56.3 (2004), pp. 221–255.

[BM92] P. J. Besl and N. D. McKay. “A method for registration of 3-D shapes.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 14.2 (1992), pp. 239–256.

[BN76] J. F. Blinn and M. E. Newell. “Texture and Reflection in Computer Generated Images.” In: *Communications of the ACM* 19.10 (1976), pp. 542–547.

[BSD08] L. Bavoil, M. Sainz, and R. Dimitrov. “Image-space horizon-based ambient occlusion.” In: *ACM SIGGRAPH Talks*. 2008, p. 22.

[Bun05] M. Bunnell. “Dynamic Ambient Occlusion and Indirect Lighting.” In: *GPU Gems 2*. Ed. by M. Pharr. 1st. Addison-Wesley Professional, 2005. Chap. 14, pp. 223–233.

[Cal+13] D. A. Calian, K. Mitchell, D. Nowrouzezahrai, and J. Kautz. “The Shading Probe: Fast Appearance Acquisition for Mobile AR.” In: *ACM SIGGRAPH Asia Technical Briefs*. 2013, 20:1–20:4.

[CBI13] J. Chen, D. Bautembach, and S. Izadi. “Scalable Real-time Volumetric Surface Reconstruction.” In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), 113:1–113:16.

[CCC08] M. Corsini, M. Callieri, and P. Cignoni. “Stereo Light Probe.” In: *Computer Graphics Forum (Proc. Eurographics)* 27.2 (2008), pp. 291–300.

[CD01] J. Cohen and P. Debevec. *LightGen: HDRShop Plugin*. 2001.

[CG12] C. Crassin and S. Green. “Octree-Based Sparse Voxelization Using the GPU Hardware Rasterizer.” In: *OpenGL Insights*. Ed. by P. Cozzi and C. Riccio. 1st. CRC Press, 2012. Chap. 22, pp. 303–319.

[Che95] S. E. Chen. “QuickTime VR: An Image-based Approach to Virtual Environment Navigation.” In: *Proceedings of SIGGRAPH ’95*. 1995, pp. 29–38.

[Chr08] P. H. Christensen. *Point-Based Approximate Color Bleeding*. Tech. rep. Pixar Animation Studios, 2008.

[CK07] M. Colbert and J. Křivánek. “GPU-based importance sampling.” In: *GPU Gems 3*. Ed. by H. Nguyen. 1st. Addison-Wesley, 2007. Chap. 20, pp. 459–476.

[CL96] B. Curless and M. Levoy. “A Volumetric Method for Building Complex Models from Range Images.” In: *Proceedings of SIGGRAPH ’96*. 1996, pp. 303–312.

[Coh+88] M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. “A Progressive Refinement Approach to Fast Radiosity Image Generation.” In: *Computer Graphics (Proc. SIGGRAPH)* 22.4 (1988), pp. 75–84.

[Coo86] R. L. Cook. “Stochastic Sampling in Computer Graphics.” In: *ACM Transactions on Graphics (TOG)* 5.1 (1986), pp. 51–72.

[CPC84] R. L. Cook, T. Porter, and L. Carpenter. “Distributed Ray Tracing.” In: *Computer Graphics (Proc. SIGGRAPH)* 18.3 (1984), pp. 137–145.

[Cra+09] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann. “GigaVoxels: Ray-guided Streaming for Efficient and Detailed Voxel Rendering.” In: *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*. 2009, pp. 15–22.

[Cra+11] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann. “Interactive Indirect Illumination Using Voxel Cone Tracing.” In: *Computer Graphics Forum (Proc. Pacific Graphics)* 30.7 (2011), pp. 1921–1930.

[Cra11] C. Crassin. “GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes.” PhD thesis. Université Grenoble, France, 2011.

[Cro77] F. C. Crow. “Shadow Algorithms for Computer Graphics.” In: *Computer Graphics (Proc. SIGGRAPH)* 11.2 (1977), pp. 242–248.

[CRY00] A. Criminisi, I. Reid, and A. Ynnerman. “Single View Metrology.” In: *International Journal of Computer Vision* 40.2 (2000), pp. 123–148.

[CSF99] A. C. Costa, A. A. Sousa, and F. N. Ferreira. “Lighting Design: A Goal Based Approach Using Optimisation.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. Springer Vienna, 1999, pp. 317–328.

[Cso+14] M. Csongei, L. Hoang, C. Sandor, and Y. B. Lee. “Global illumination for Augmented Reality on mobile phones.” In: *IEEE Virtual Reality (VR) Posters*. 2014, pp. 69–70.

[CT82] R. L. Cook and K. E. Torrance. “A reflectance model for computer graphics.” In: *ACM Transactions on Graphics (TOG)* 1.1 (1982), pp. 7–24.

[Dac+14] C. Dachsbacher, J. Křivánek, M. Hašan, A. Arbree, B. Walter, and J. Novák. “Scalable Realistic Rendering with Many-Light Methods.” In: *Computer Graphics Forum (also Proc. Eurographics –State of the Art Reports)* 33.1 (2014), pp. 88–104.

[Dai+17] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. “BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration.” In: *ACM Transactions on Graphics (TOG)* 36.3 (2017), 24:1–24:18.

[Dan+99] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. “Reflectance and Texture of Real-world Surfaces.” In: *ACM Transactions on Graphics (TOG)* 18.1 (1999), pp. 1–34.

[Dav+07] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. “MonoSLAM: Real-Time Single Camera SLAM.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 29.6 (2007), pp. 1052–1067.

[Dav99] A. Davenport. *The History of Photography: An Overview*. 2nd. Albuquerque: University of New Mexico Press, 1999.

[DC99] T. Drummond and R. Cipolla. “Real-Time Tracking of Complex Structures With on-Line Camera Calibration.” In: *Proc. British Machine Vision Conference (BMVC)*. 1999, pp. 574–583.

[Deb+12] P. Debevec, P. Graham, J. Busch, and M. Bolas. “A Single-shot Light Probe.” In: *ACM SIGGRAPH Talks*. 2012, 10:1–10:1.

[Deb98] P. Debevec. “Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography.” In: *Proceedings of SIGGRAPH ’98*. 1998, pp. 189–198.

[DL06] W. Donnelly and A. Lauritzen. “Variance Shadow Maps.” In: *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*. 2006, pp. 161–165.

[DLW93] P. Dutré, E. P. F. Lafortune, and Y. D. Willems. “Monte Carlo Light Tracing with Direct Computation of Pixel Intensities.” In: *Proc. International Conference on Computational Graphics and Visualization Techniques (Compugraphics)*. 1993, pp. 128–137.

[DM97] P. Debevec and J. Malik. “Recovering High Dynamic Range Radiance Maps from Photographs.” In: *Proceedings of SIGGRAPH ’97*. 1997, pp. 369–378.

[Don+09] Z. Dong, T. Grosch, T. Ritschel, J. Kautz, and H.-P. Seidel. “Real-time Indirect Illumination with Clustered Visibility.” In: *Proc. Vision, Modeling and Visualization (VMV)*. Vol. 9. 2009, pp. 187–196.

[Don+14] Y. Dong, G. Chen, P. Peers, J. Zhang, and X. Tong. “Appearance-from-motion: Recovering Spatially Varying Surface Reflectance Under Unknown Lighting.” In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), 193:1–193:12.

[DRB97] G. Drettakis, L. Robert, and S. Bougnoux. “Interactive Common Illumination for Computer Augmented Reality.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 1997, pp. 45–56.

[DS05] C. Dachsbacher and M. Stamminger. “Reflective Shadow Maps.” In: *Proc. Symposium on Interactive 3D Graphics and Games (I3D)*. 2005, pp. 203–231.

[DS06] C. Dachsbacher and M. Stamminger. “Splatting Indirect Illumination.” In: *Proc. Symposium on Interactive 3D Graphics and Games (I3D)*. 2006, pp. 93–100.

[DW00] J. M. DiCarlo and B. A. Wandell. “Rendering high dynamic range images.” In: *Proc. SPIE, Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications*. Vol. 3965. 2000, pp. 392–401.

[ED08] E. Eisemann and X. Décoret. “Single-pass GPU Solid Voxelization for Real-time Applications.” In: *Proc. Graphics Interface (GI)*. 2008, pp. 73–80.

[Eil+17] G. Eilertsen, Kronander Joel, G. Denes, and J. Mantiuk Rafałand Unger. “HDR image reconstruction from a single exposure using deep CNNs.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 36.6 (2017), 178:1–178–15.

[Eng09] W. Engel. “Advances in Real-Time Rendering in Games: The Light Pre-Pass Renderer: Renderer Design for Efficient Support of Multiple Lights.” In: *ACM SIGGRAPH Courses*. 2009, 4:1–4:42.

[Fau93] O. D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT press, 1993.

[FB81] M. A. Fischler and R. C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[FBS05] J. Fischer, D. Bartz, and W. Straber. "Stylized augmented reality for improved immersion." In: *Proc. IEEE Virtual Reality (VR)*. 2005, pp. 195–202.

[Fer01] J. A. Ferwerda. "Elements of Early Vision for Computer Graphics." In: *IEEE Computer Graphics and Applications (CGA)* 21.5 (2001), pp. 22–33.

[FGR93] A. Fournier, A. S. Gunavan, and C. Romanzin. "Common Illumination between Real and Computer Generated Scenes." In: *Proc. Graphics Interface (GI)*. 1993, pp. 254–262.

[FJ08a] T. A. Franke and Y. Jung. "Real-Time Mixed Reality with GPU Techniques." In: *Proc. International Conference on Computer Graphics Theory and Applications (GRAPP)*. 2008, pp. 249–252.

[FJ08b] T. Franke and Y. Jung. "Precomputed Radiance Transfer for X3D based Mixed Reality Applications." In: *Proc. International Symposium on 3D Web Technology (Web3D)*. 2008, pp. 7–10.

[Fra13a] T. A. Franke. "Delta Light Propagation Volumes for Mixed Reality." In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2013, pp. 125–132.

[Fra13b] T. A. Franke. "Near-Field Illumination for Mixed Reality with Delta Radiance Fields." In: *ACM SIGGRAPH Posters*. 2013, 76:1–76:1.

[Fra14a] T. A. Franke. "Delta Voxel Cone Tracing." In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2014, pp. 39–44.

[Fra14b] T. A. Franke. "Interactive Relighting of Arbitrary Rough Surfaces." In: *ACM SIGGRAPH Posters*. 2014, 28:1–28:1.

[Fra15] T. A. Franke. "The Delta Radiance Field." PhD thesis. Technische Universität Darmstadt, Germany, 2015.

[FZC93] G. W. Fitzmaurice, S. Zhai, and M. H. Chignell. "Virtual Reality for Palmtop Computers." In: *ACM Transactions on Information Systems (TOIS)* 11.3 (1993), pp. 197–218.

[Gar+12] E. Garces, A. Munoz, J. Lopez-Moreno, and D. Gutierrez. "Intrinsic Images by Clustering." In: *Computer Graphics Forum (Proc. EGSR)* 31.4 (2012), pp. 1415–1424.

[Gar+14] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. "Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion." In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292.

[GEM07] T. Grosch, T. Eble, and S. Müller. "Consistent Interactive Augmentation of Live Camera Images with Correct Near-field Illumination." In: *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST)*. 2007, pp. 125–132.

[Geo+12] I. Georgiev, J. Křivánek, T. Davidovič, and P. Slusallek. "Light Transport Simulation with Vertex Connection and Merging." In:

ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31.6 (2012), 192:1–192:10.

[Ger14] T. Gerrits. “Untersuchung von Methoden zum automatischen Einbetten fotorealistischer Objekte in Szenen ohne zusätzlichen Szeneninformationen.” Bachelor’s thesis. Otto-von-Guericke-Universität Magdeburg, Germany, 2014.

[GHH01] S. Gibson, T. Howard, and R. Hubbold. “Flexible Image-Based Photometric Reconstruction using Virtual Light Sources.” In: *Computer Graphics Forum (Proc. Eurographics)* 20.3 (2001), pp. 203–214.

[Gib+03] S. Gibson, J. Cook, T. Howard, and R. Hubbold. “Rapid Shadow Generation in Real-world Lighting Environments.” In: *Proc. Eurographics Symposium on Rendering (EGSR)*. 2003, pp. 219–229.

[GM00] S. Gibson and A. Murta. “Interactive Rendering with Real-World Illumination.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 2000, pp. 365–376.

[GN04] M. D. Grossberg and S. K. Nayar. “Modeling the Space of Camera Response Functions.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 26.10 (2004), pp. 1272–1282.

[Gor+84] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. “Modeling the Interaction of Light Between Diffuse Surfaces.” In: *Computer Graphics (Proc. SIGGRAPH)* 18.3 (1984), pp. 213–222.

[Gor+96] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. “The Lumigraph.” In: *Proceedings of SIGGRAPH ’96*. 1996, pp. 43–54.

[Gre+98] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg. “The Irradiance Volume.” In: *IEEE Computer Graphics and Applications (CGA)* 18.2 (1998), pp. 32–43.

[Gre03] R. Green. “Spherical Harmonic Lighting: The Gritty Details.” In: *Game Developers Conference (GDC)*. 2003, pp. 1–47.

[Gro+09] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. “Ground truth dataset and baseline evaluations for intrinsic image algorithms.” In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2009, pp. 2335–2342.

[Gro05a] T. Grosch. “Differential Photon Mapping - Consistent Augmentation of Photographs with Correction of all Light Paths.” In: *Proc. Eurographics Short Presentations*. 2005, pp. 53–56.

[Gro05b] T. Grosch. “PanoAR: Interactive Augmentation of Omni-directional Images with Consistent Lighting.” In: *Proc. Computer Vision / Computer Graphics Collaboration Techniques and Applications (Mirage)*. 2005, pp. 25–34.

[Gro07] T. Grosch. “Augmentierte Bildsynthese.” PhD thesis. Universität Koblenz-Landau, Germany, 2007.

[GRS12] L. Gruber, T. Richter-Trummer, and D. Schmalstieg. “Real-time Photometric Registration from Arbitrary Geometry.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2012, pp. 119–128.

[Gru+14] L. Gruber, T. Langlotz, P. Sen, T. Höherer, and D. Schmalstieg. “Efficient and robust radiance transfer for probeless photorealistic augmented reality.” In: *Proc. IEEE Virtual Reality (VR)*. 2014, pp. 15–20.

[Gua+16] D. Guarnera, G. C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. "BRDF Representation and Acquisition." In: *Computer Graphics Forum (CGF)* 35.2 (2016), pp. 625–650.

[Gui+00] E. Guillou, D. Meneveaux, E. Maisel, and K. Bouatouch. "Using vanishing points for camera calibration and coarse 3D reconstruction from a single image." In: *The Visual Computer* 16.7 (2000), pp. 396–410.

[GVS15] L. Gruber, J. Ventura, and D. Schmalstieg. "Image-space illumination for augmented reality in dynamic environments." In: *Proc. IEEE Virtual Reality (VR)*. 2015, pp. 127–134.

[HAO05] J. Hasselgren, T. Akenine-Möller, and L. Ohlsson. "Conservative Rasterization." In: *GPU Gems 2*. Ed. by M. Pharr. 1st. Addison-Wesley Professional, 2005. Chap. 42, pp. 677–690.

[Har97] R. I. Hartley. "In Defense of the Eight-Point Algorithm." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 19.6 (1997), pp. 580–593.

[Haš+09] M. Hašan, J. Křivánek, B. Walter, and K. Bala. "Virtual Spherical Lights for Many-light Rendering of Glossy Scenes." In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 28.5 (2009), 143:1–143:6.

[Hav+05a] V. Havran, M. Smyk, G. Krawczyk, K. Myszkowski, and H.-P. Seidel. "Importance Sampling for Video Environment Maps." In: *ACM SIGGRAPH Sketches*. 2005.

[Hav+05b] V. Havran, M. Smyk, G. Krawczyk, K. Myszkowski, and H.-P. Seidel. "Interactive System for Dynamic Scene Lighting using Captured Video Environment Maps." In: *Proc. Eurographics Symposium on Rendering (EGSR)*. 2005, pp. 31–42.

[Hav+17] V. Havran, J. Hošek, Š. Němcová, J. Čáp, and J. Bittner. "Lightdrum - Portable Light Stage for Accurate BTF Measurement on Site." In: *Sensors* 17.3 (2017), 423:1–423:57.

[HDH03] M. Haller, S. Drab, and W. Hartmann. "A Real-time Shadow Approach for an Augmented Reality Application Using Shadow Volumes." In: *Proc. ACM Symposium on Virtual Reality Software and Technology (VRST)*. 2003, pp. 56–65.

[HEH05] D. Hoiem, A. A. Efros, and M. Hebert. "Automatic Photo Pop-up." In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 24.3 (2005), pp. 577–584.

[Hen+12] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments." In: *International Journal of Robotics Research* 31.5 (2012), pp. 647–663.

[Heu98] F. A. van den Heuvel. "3D reconstruction from a single image using geometric constraints." In: *Journal of Photogrammetry and Remote Sensing (ISPRS)* 53.6 (1998), pp. 354–368.

[HHF09] V. Hedau, D. Hoiem, and D. Forsyth. "Recovering the Spatial Layout of Cluttered Rooms." In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2009, pp. 1849–1856.

[HJ09] T. Hachisuka and H. W. Jensen. "Stochastic Progressive Photon Mapping." In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28.5 (2009), 141:1–141:8.

[HJ10] T. Hachisuka and H. W. Jensen. "Parallel Progressive Photon Mapping on GPUs." In: *ACM SIGGRAPH Asia Sketches*. 2010, p. 54.

[HMY12] T. Harada, J. McKee, and J. C. Yang. “Forward+: Bringing Deferred Lighting to the Next Level.” In: *Proc. Eurographics Short Papers*. 2012, pp. 1–4.

[Hof+10] N. Hoffman, Y. Gotanda, A. Martinez, and B. Snow. “Physically-Based Shading Models in Film and Game Production.” In: *ACM SIGGRAPH Courses*. 2010.

[HOJ08] T. Hachisuka, S. Ogaki, and H. W. Jensen. “Progressive Photon Mapping.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 27.5 (2008), 130:1–130:8.

[Hol+11] M. Holländer, T. Ritschel, E. Eisemann, and T. Boubekeur. “ManyLoDs: Parallel Many-View Level-of-Detail Selection for Real-Time Global Illumination.” In: *Computer Graphics Forum (Proc. EGSR)* 30.4 (2011), pp. 1233–1240.

[HPJ12] T. Hachisuka, J. Pantaleoni, and H. W. Jensen. “A Path Space Extension for Robust Light Transport Simulation.” In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), 191:1–191:10.

[HSA91] P. Hanrahan, D. Salzman, and L. Aupperle. “A Rapid Hierarchical Radiosity Algorithm.” In: *Computer Graphics (Proc. SIGGRAPH)* 25.4 (1991), pp. 197–206.

[HSO07] M. Harris, S. Sengupta, and J. D. Owens. “Parallel Prefix Sum (Scan) with CUDA.” In: *GPU Gems 3*. Ed. by H. Nguyen. 1st. Addison-Wesley, 2007. Chap. 39, pp. 851–876.

[Hu+00] H. H. Hu, A. A. Gooch, W. B. Thompson, B. E. Smits, J. J. Rieser, and P. Shirley. “Visual Cues for Imminent Object Contact in Realistic Virtual Environment.” In: *Proc. IEEE Visualization (VIS)*. 2000, pp. 179–185.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. 2nd. Cambridge University Press, 2004.

[IB98] M. Isard and A. Blake. “CONDENSATION - conditional density propagation for visual tracking.” In: *International Journal of Computer Vision* 29.1 (1998), pp. 5–28.

[IHS17] Y. Itoh, T. Hamasaki, and M. Sugimoto. “Occlusion Leak Compensation for Optical See-Through Displays Using a Single-Layer Transmissive Spatial Light Modulator.” In: *IEEE Transactions on Visualization and Computer Graphics (Proc. ISMAR)* 23.11 (2017), pp. 2463–2473.

[Int96] International Commission on Illumination. *The Basis of Physical Photometry*. 2nd ed. CIE publications. Commission Internationale de l’Éclairage, 1996.

[Iza+11] S. Izadi et al. “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera.” In: *Proc. ACM Symposium on User Interface Software and Technology*. 2011, pp. 559–568.

[Jac+05] K. Jacobs, J.-D. Nahmias, C. Angus, A. Reche, C. Loscos, and A. Steed. “Automatic Generation of Consistent Shadows for Augmented Reality.” In: *Proc. Graphics Interface (GI)*. 2005, pp. 113–120.

[Jen+01] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan. “A Practical Model for Subsurface Light Transport.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH)*. 2001, pp. 511–518.

[Jen01] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. 1st. A K Peters, Ltd., 2001.

[Jen96] H. W. Jensen. “Global Illumination Using Photon Maps.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 1996, pp. 21–30.

[JL06] K. Jacobs and C. Loscos. “Classification of Illumination Methods for Mixed Reality.” In: *Computer Graphics Forum (Proc. Eurographics)* 25.1 (2006), pp. 29–51.

[JND12] J. Jachnik, R. A. Newcombe, and A. J. Davison. “Real-time Surface Light-field Capture for Augmentation of Planar Specular Surfaces.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2012, pp. 91–97.

[Joh03] C. S. Johnson. *Rectilinear Wide-angle Lenses and Fish-Eye (All Sky) Lenses*. Tech. rep. University of North Carolina at Chapel Hill, 2003, pp. 1–6.

[JP16] M. Jelonek and M. Prilla. “Motivational Aspects of Using Augmented Reality Glasses in Care.” In: *Mensch und Computer Workshop*. 2016, pp. 1–6.

[JRM16] S. Jiddi, P. Robert, and E. Marchand. “Reflectance and Illumination Estimation for Realistic Augmentations of Real Scenes.” In: *IEEE ISMAR Posters*. 2016, pp. 1–6.

[JRM17] S. Jiddi, P. Robert, and E. Marchand. “Illumination Estimation using Cast Shadows for Realistic Augmented Reality Applications.” In: *IEEE ISMAR Posters*. 2017.

[Jua+05] M. C. Juan, M. Alcañiz, C. Monserrat, C. Botella, R. M. Baños, and B. Guerrero. “Using Augmented Reality to Treat Phobias.” In: *IEEE Computer Graphics and Applications (CGA)* 25.6 (2005), pp. 31–37.

[Käh+15] O. Kähler, V. Adrian Prisacariu, C. Yuheng Ren, X. Sun, P. Torr, and D. Murray. “Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices.” In: *IEEE Transactions on Visualization and Computer Graphics (Proc. ISMAR)* 21.11 (2015), pp. 1241–1250.

[Kaj86] J. T. Kajiya. “The Rendering Equation.” In: *Computer Graphics (Proc. SIGGRAPH)* 20.4 (1986), pp. 143–150.

[Kan+01] H. W. Kang, S. H. Pyo, K.-i. Anjyo, and S. Y. Shin. “Tour Into the Picture using a Vanishing Line and its Extension to Panoramic Images.” In: *Computer Graphics Forum (Proc. Eurographics)* 20.3 (2001), pp. 132–141.

[Kán15] P. Kán. “Interactive HDR Environment Map Capturing on Mobile Devices.” In: *Proc. Eurographics Short Papers*. 2015, pp. 29–32.

[Kap09] A. Kaplanyan. “Light Propagation Volumes in CryEngine 3.” In: *ACM SIGGRAPH Courses*. 2009, pp. 1–45.

[Kar+11] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem. “Rendering Synthetic Objects into Legacy Photographs.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 30.6 (2011), 157:1–157:12.

[Kau06] J. Kautz. “Physically-Based Reflectance for Games: Reflectance Rendering with Environment Map Lighting.” In: *ACM SIGGRAPH Courses*. 2006, pp. 1–95.

[KB04] L. Kobbelt and M. Botsch. “A Survey of Point-Based Techniques in Computer Graphics.” In: *Computers & Graphics* 28.6 (2004), pp. 801–814.

[KBH06] M. Kazhdan, M. Bolitho, and H. Hoppe. “Poisson Surface Reconstruction.” In: *Eurographics Symposium on Geometry Processing (SGP)*. 2006, pp. 61–70.

[KC08] J. Křivánek and M. Colbert. “Real-time Shading with Filtered Importance Sampling.” In: *Computer Graphics Forum (Proc. EGSR)* 27.4 (2008), pp. 1147–1154.

[KD10] A. Kaplanyan and C. Dachsbacher. “Cascaded Light Propagation Volumes for Real-time Indirect Illumination.” In: *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*. 2010, pp. 99–107.

[Kel+13] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. “Real-time 3D Reconstruction in Dynamic Scenes using Point-based Fusion.” In: *Proc. International Conference on 3D Vision (3DV)*. 2013, pp. 1–8.

[Kel97] A. Keller. “Instant Radiosity.” In: *Proceedings of SIGGRAPH ’97*. 1997, pp. 49–56.

[KH13] M. Kazhdan and H. Hoppe. “Screened Poisson Surface Reconstruction.” In: *ACM Transactions on Graphics (TOG)* 32.3 (2013), 29:1–29:13.

[KK12a] P. Kán and H. Kaufmann. “High-Quality Reflections, Refractions, and Caustics in Augmented Reality and their Contribution to Visual Coherence.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2012, pp. 99–108.

[KK12b] P. Kán and H. Kaufmann. “Physically-Based Depth of Field in Augmented Reality.” In: *Proc. Eurographics Short Papers*. 2012, pp. 89–92.

[KK13a] P. Kán and H. Kaufmann. “Differential Irradiance Caching for fast high-quality light transport between virtual and real worlds.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2013, pp. 133–141.

[KK13b] P. Kán and H. Kaufmann. “Differential Progressive Path Tracing for High-Quality Previsualization and Relighting in Augmented Reality.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2013, pp. 133–141.

[KK14] S. B. Knorr and D. Kurz. “Real-time illumination estimation from faces for coherent rendering.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2014, pp. 113–122.

[KK15] P. Kán and H. Kaufmann. “Mobile Multiview Diffuse Texture Extraction.” In: *Proc. Smart Tools and Apps in Computer Graphics (STAG)*. 2015, pp. 113–120.

[KKS17] P. Klimant, C. Kollatsch, and M. Schumann. “Augmented Reality Solutions in Mechanical Engineering.” In: *Proc. ASME International Manufacturing Science and Engineering Conference (MSEC)*. American Society of Mechanical Engineers. 2017, pp. 1–9.

[KM06] G. Klein and D. W. Murray. “Full-3D Edge Tracking with a Particle Filter.” In: *Proc. British Machine Vision Conference (BMVC)*. 2006, pp. 1119–1128.

[KM07] G. Klein and D. Murray. “Parallel Tracking and Mapping for Small AR Workspaces.” In: *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. 2007, pp. 225–234.

[KM08] G. Klein and D. Murray. “Compositing for Small Cameras.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2008, pp. 57–60.

[KMS07] D. Kalkofen, E. Mendez, and D. Schmalstieg. “Interactive Focus and Context Visualization for Augmented Reality.” In: *IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. 2007, pp. 1–10.

[Kne+10] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer. “Differential Instant Radiosity for Mixed Reality.” In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2010, pp. 99–107.

[Kne+11] M. Knecht, C. Traxler, W. Purgathofer, and M. Wimmer. “Adaptive Camera-Based Color Mapping For Mixed-Reality Applications.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2011, pp. 165–168.

[Kne+12] M. Knecht, C. Traxler, O. Mattausch, and M. Wimmer. “Augmented Reality: Reciprocal Shading for Mixed Reality.” In: *Computers & Graphics* 36.7 (2012), pp. 846–856.

[Kne+13] M. Knecht, C. Traxler, C. Winklhofer, and M. Wimmer. “Reflective and Refractive Objects for Mixed Reality.” In: *IEEE Transactions on Visualization and Computer Graphics (Proc. VR)* 19.4 (2013), pp. 576–582.

[Kol+17] C. Kollatsch, M. Schumann, P. Klimant, and M. Lorenz. “Industrial Augmented Reality: Transferring a Numerical Control connected Augmented Realty System from Marketing to Maintenance.” In: *IEEE ISMAR Posters*. 2017.

[Kor+07] M. Korn, M. Stange, A. von Arb, L. Blum, M. Kreil, K.-J. Kunze, J. Anhenn, T. Wallrath, and T. Grosch. “Interactive Augmentation of Live Images using a HDR Stereo Camera.” In: *Journal of Virtual Reality and Broadcasting (JVRB)* 4.12 (2007), pp. 107–118.

[Kro+14] J. Kronander, S. Gustavson, G. Bonnet, A. Ynnerman, and J. Unger. “A Unified Framework for Multi-Sensor HDR Video Reconstruction.” In: *Signal Processing: Image Communication* 29.2 (2014), pp. 203–215.

[Kro+15] J. Kronander, F. Banterle, A. Gardner, E. Miandji, and J. Unger. “Photorealistic rendering of mixed reality scenes.” In: *Computer Graphics Forum (Proc. Eurographics)* 34.2 (2015), pp. 643–665.

[KSO10] L. Kavan, P.-P. Sloan, and C. O’Sullivan. “Fast and Efficient Skinning of Animated Meshes.” In: *Computer Graphics Forum (CGF)* 29.2 (2010), pp. 327–336.

[KY02] M. Kanbara and N. Yokoya. “Geometric and Photometric Registration for Real-Time Augmented Reality.” In: *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. 2002, pp. 279–280.

[KZ11] C. Knaus and M. Zwicker. “Progressive Photon Mapping: A Probabilistic Approach.” In: *ACM Transactions on Graphics (TOG)* 30.3 (2011), 25:1–25:13.

[Laf+97] E. P. F. Lafortune, S.-C. Foo, K. E. Torrance, and D. P. Greenberg. “Non-linear Approximation of Reflectance Functions.” In: *Proceedings of SIGGRAPH ’97*. 1997, pp. 117–126.

[Lau10] A. Lauritzen. “Beyond Programmable Shading: Deferred Rendering for Current and Future Rendering Pipelines.” In: *ACM SIGGRAPH Courses*. 2010, pp. 1–34.

[LB12] P. Lensing and W. Broll. “Instant indirect illumination for dynamic mixed reality scenes.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2012, pp. 109–118.

[LB13] P. Lensing and W. Broll. “LightSkin: Real-time Global Illumination for Virtual and Mixed Reality.” In: *Proc. Joint Virtual Reality Conference (JVRC)*. Aire-la-Ville, Switzerland, Switzerland, 2013, pp. 17–24.

[LC87] W. E. Lorensen and H. E. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm.” In: *Computer Graphics (Proc. SIGGRAPH)* 21.4 (1987), pp. 163–169.

[LCZ99] D. Liebowitz, A. Criminisi, and A. Zisserman. “Creating Architectural Models from Images.” In: *Computer Graphics Forum (Proc. Eurographics)*. Vol. 18. 1999, pp. 39–50.

[LE07] J.-F. Lalonde and A. A. Efros. “Using Color Compatibility for Assessing Image Realism.” In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2007, pp. 1–8.

[Len+01] H. P. A. Lensch, J. Kautz, M. Goesele, W. Heidrich, and H.-P. Seidel. “Image-based Reconstruction of Spatially Varying Materials.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 2001, pp. 103–114.

[Len03] H. P. A. Lensch. “Efficient, image-based appearance acquisition of real-world objects.” PhD thesis. Universität des Saarlandes, Germany, 2003.

[LEN09] J.-F. Lalonde, A. A. Efros, and S. G. Narasimhan. “Webcam Clip Art: Appearance and Illuminant Transfer from Time-lapse Sequences.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 28.5 (2009), 131:1–131:10.

[Lew94] R. R. Lewis. “Making shaders more physically plausible.” In: *Computer Graphics Forum (CGF)* 13.2 (1994), pp. 109–120.

[LH06] H. Li and R. I. Hartley. “Five-Point Motion Estimation Made Easy.” In: *International Conference on Pattern Recognition (ICPR)*. Vol. 1. 2006, pp. 630–633.

[LH96] M. Levoy and P. Hanrahan. “Light Field Rendering.” In: *Proceedings of SIGGRAPH ’96*. 1996, pp. 31–42.

[Liu+04] X. Liu, P.-P. Sloan, H.-Y. Shum, and J. Snyder. “All-Frequency Precomputed Radiance Transfer for Glossy Objects.” In: *Proc. Eurographics Symposium on Rendering (EGSR)*. 2004, pp. 337–344.

[Liu+10] C. Liu, L. Sharan, E. H. Adelson, and R. Rosenholtz. “Exploring Features in a Bayesian Framework for Material Recognition.” In: *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 239–246.

[Lon81] H. C. Longuet-Higgins. “A computer algorithm for reconstructing a scene from two projections.” In: *Nature* 293 (1981), pp. 133–135.

[Los+99] C. Loscos, M.-C. Frasson, G. Drettakis, B. Walter, X. Granier, and P. Poulin. “Interactive Virtual Relighting and Remodeling of Real Scenes.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 1999, pp. 329–340.

[Löw+09] J. Löw, A. Ynnerman, P. Larsson, and J. Unger. “HDR Light Probe Sequence Resampling for Realtime Incident Light Field Rendering.” In: *Proc. Spring Conference on Computer Graphics (SCCG)*. 2009, pp. 43–50.

[LS05] I. Lazániy and L. Szirmay-Kalos. “Fresnel Term Approximations for Metals.” In: *Proc. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)* (2005), pp. 77–80.

[LTG93] D. Lischinski, F. Tampieri, and D. P. Greenberg. “Combining Hierarchical Radiosity and Discontinuity Meshing.” In: *Proceedings of SIGGRAPH ’93*. 1993, pp. 199–208.

[LW85] M. Levoy and T. Whitted. *The use of points as a display primitive*. Tech. rep. University of North Carolina, Department of Computer Science, 1985, pp. 1–19.

[LW93] E. P. F. Lafortune and Y. D. Willems. “Bi-directional Path Tracing.” In: *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics ’93)*. 1993, pp. 145–153.

[Man+17] D. Mandl, K. M. Yi, P. Mohr, P. Roth, P. Fua, V. Lepetit, D. Schmalstieg, and D. Kalkofen. “Learning Lightprobes for Mixed Reality Illumination.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2017.

[Mar98] S. R. Marschner. “Inverse Rendering in Computer Graphics.” PhD thesis. Cornell University, 1998.

[Mat+03] W. Matusik, H. Pfister, M. Brand, and L. McMillan. “A Data-driven Reflectance Model.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 22.3 (2003), pp. 759–769.

[MBC13] M. Meilland, C. Barat, and A. Comport. “3D High Dynamic Range dense visual SLAM and its application to real-time object re-lighting.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2013, pp. 143–152.

[McA+99] D. K. McAllister, L. Nyland, V. Popescu, A. Lastra, and C. McCue. “Real-Time Rendering of Real World Environments.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 1999, pp. 145–160.

[McK12] J. McKee. “Technology Behind AMD’s "Leo Demo".” In: *Proc. Game Developers Conference (GDC)*. 2012, pp. 1–25.

[MG97] S. R. Marschner and D. P. Greenberg. “Inverse Lighting for Photography.” In: *Proc. Color and Imaging Conference*. 1997, pp. 262–265.

[MH84] G. S. Miller and C. R. Hoffman. “Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments.” In: *ACM SIGGRAPH Courses*. 1984, pp. 1–12.

[Mil94] G. Miller. “Efficient Algorithms for Local and Global Accessibility Shading.” In: *Proceedings of SIGGRAPH ’94*. 1994, pp. 319–326.

[Mit07] M. Mittring. “Advanced in Real-Time Rendering in 3D Graphics and Games: Finding next gen: Cryengine 2.” In: *ACM SIGGRAPH Courses*. 2007, pp. 97–121.

[Mit12] M. Mittring. “Advances in Real-Time Rendering in 3D Graphics and Games: The Technology Behind the 3D Graphics and Games Course “Unreal Engine 4 Elemental demo”.” In: *ACM SIGGRAPH Courses*. 2012, pp. 1–71.

[MK94] P. Milgram and F. Kishino. “A Taxonomy of Mixed Reality Visual Displays.” In: *IEICE Transactions on Information and Systems* 77.12 (1994), pp. 1321–1329.

[MKC07] R. Marroquim, M. Kraus, and P. R. Cavalcanti. “Efficient Point-Based Rendering Using Image Reconstruction.” In: *Proc. Symposium on Point-Based Graphics*. 2007, pp. 101–108.

[ML11] C. B. Madsen and B. B. Lal. “Outdoor Illumination Estimation in Image Sequences for Augmented RealityNo Title.” In: *Proc. International Conference on Computer Graphics Theory and Applications (GRAPP)*. 2011, pp. 129–139.

[MLH02] D. K. McAllister, A. Lastra, and W. Heidrich. “Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions.” In: *Proc. ACM SIGGRAPH/Eurographics Conference on Graphics Hardware (HWWS)*. 2002, pp. 79–88.

[MLJ09] X. Mei, H. Ling, and D. W. Jacobs. “Sparse Representation of Cast Shadows via l1-Regularized Least Squares.” In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2009, pp. 583–590.

[MN08] C. B. Madsen and M. Nielsen. “Towards Probe-less Augmented Reality - A Position Paper.” In: *Proc. International Conference on Computer Graphics Theory and Applications (GRAPP)*. 2008, pp. 255–261.

[MN99] T. Mitsunaga and S. K. Nayar. “Radiometric Self Calibration*.” In: *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 1999, pp. 374–380.

[MTB16] A. Morgand, M. Tamaazousti, and A. Bartoli. “A Geometric Model for Specularity Prediction on Planar Surfaces with Multiple Light Sources.” In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2016), to appear.

[MTB17] A. Morgand, M. Tamaazousti, and A. Bartoli. “A Multiple-View Geometric Model of Specularities on Non-Planar Shapes with Application to Dynamic Retexturing.” In: *IEEE Transactions on Visualization and Computer Graphics (Proc. ISMAR)* 23.11 (2017), pp. 2485–2493.

[Mül95] V. Müller. “Polarization-Based Separation of Diffuse and Specular Surface-Reflection.” In: *Proc. Mustererkennung: Verstehen akustischer und visueller Informationen*. 1995, pp. 202–209.

[Nak+86] E. Nakamae, K. Harada, T. Ishizaki, and T. Nishita. “A Montage Method: The Overlaying of the Computer Generated Images Onto a Background Photograph.” In: *Computer Graphics (Proc. SIGGRAPH)* 20.4 (1986), pp. 207–214.

[New+11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. “KinectFusion: Real-time Dense Surface Mapping and Tracking.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2011, pp. 127–136.

[Nie+13] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. “Real-time 3D Reconstruction at Scale using Voxel Hashing.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 32.6 (2013), 169:1–169:11.

[Nis04] D. Nistér. “An Efficient Solution to the Five-Point Relative Pose Problem.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 26.6 (2004), pp. 756–770.

[NLD11] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. “DTAM: Dense Tracking and Mapping in Real-Time.” In: *Proc. International Conference on Computer Vision (ICCV)*. 2011, pp. 2320–2327.

[NNB04] D. Nistér, O. Naroditsky, and J. Bergen. “Visual Odometry.” In: *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Ieee. 2004, pp. 652–659.

[Now+11] D. Nowrouzezahrai, S. Geiger, K. Mitchell, R. Sumner, W. Jarosz, and M. Gross. “Light Factorization for Mixed-Frequency Shadows in Augmented Reality.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2011, pp. 173–179.

[NRH03] R. Ng, R. Ramamoorthi, and P. Hanrahan. “All-frequency Shadows Using Non-linear Wavelet Lighting Approximation.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 22.3 (2003), pp. 376–381.

[NW09] G. Nichols and C. Wyman. “Multiresolution Splatting for Indirect Illumination.” In: *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*. 2009, pp. 83–90.

[Nyl98] L. Nyland. *Capturing Dense Environmental Range Information with a Panning, Scanning Laser Rangefinder*. Tech. rep. Chapel Hill, NC, USA: University of North Carolina, 1998, pp. 1–9.

[OA11] O. Olsson and U. Assarsson. “Tiled Shading.” In: *Journal of Graphics, GPU, and Game Tools (JGT)* 15.4 (2011), pp. 235–251.

[Oat05] C. Oat. “Irradiance Volumes for Games.” In: *Game Developers Conference (GDC)*. 2005, pp. 1–54.

[Oat06] C. Oat. “Irradiance Volumes for Real-time Rendering.” In: *ShaderX5: Advanced Rendering Techniques*. Ed. by W. Engel. 1st. Charles River Media, 2006. Chap. 6.1, pp. 333–344.

[OBA12a] O. Olsson, M. Billeter, and U. Assarsson. “Clustered Deferred and Forward Shading.” In: *Proc. High Performance Graphics (HPG)*. 2012, pp. 87–96.

[OBA12b] O. Olsson, M. Billeter, and U. Assarsson. “Tiled and Clustered Forward Shading.” In: *ACM SIGGRAPH Talks*. 2012.

[ODJ04] V. Ostromoukhov, C. Donohue, and P.-M. Jodoin. “Fast Hierarchical Importance Sampling with Blue Noise Properties.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 23.3 (2004), pp. 488–495.

[Oh+01] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. “Image-based Modeling and Photo Editing.” In: *Proceedings of SIGGRAPH ’01*. 2001, pp. 433–442.

[OKY06] B. Okumura, M. Kanbara, and N. Yokoya. “Augmented reality based on estimation of defocusing and motion blurring from captured images.” In: *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. 2006, pp. 219–225.

[ON94] M. Oren and S. K. Nayar. “Generalization of Lambert’s Reflectance Model.” In: *Proceedings of SIGGRAPH ’94*. 1994, pp. 239–246.

[Pes+10] S. Pessoa, G. Moura, J. Lima, V. Teichrieb, and J. Kelner. “Photorealistic rendering for augmented reality: A global illumination and brdf solution.” In: *Proc. IEEE Virtual Reality (VR)*. 2010, pp. 3–10.

[PF92] P. Poulin and A. Fournier. “Lights from Highlights and Shadows.” In: *Proc. Symposium on Interactive 3D Graphics (I3D)*. 1992, pp. 31–38.

[Pfi+00] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. “Surfels: Surface Elements As Rendering Primitives.” In: *Proceedings of SIGGRAPH ’00*. 2000, pp. 335–342.

[Pil+06] J. Pilet, A. Geiger, P. Lagger, V. Lepetit, and P. Fua. “An All-in-one Solution to Geometric and Photometric Calibration.” In: *Proc. IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*. 2006, pp. 69–78.

[PJH16] M. Pharr, W. Jakob, and G. Humphreys. *Physically-based Rendering: From Theory to Implementation*. 3rd. Morgan Kaufmann, 2016.

[POF98] P. Poulin, M. Ouimet, and M.-C. Frasson. “Interactively Modeling with Photogrammetry.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 1998, pp. 93–104.

[PP03] G. Patow and X. Pueyo. “A Survey of Inverse Rendering Problems.” In: *Computer Graphics Forum (CGF)* 22.4 (2003), pp. 663–687.

[RBS03] M. A. Robertson, S. Borman, and R. L. Stevenson. “Estimation-theoretic approach to dynamic range enhancement using multiple exposures.” In: *Journal of Electronic Imaging* 12.2 (2003), pp. 219–228.

[RD06] E. Rosten and T. Drummond. “Machine learning for high-speed corner detection.” In: *Proc. European Conference on Computer Vision (ECCV)*. 2006, pp. 430–443.

[Rei+01] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. “Color Transfer Between Images.” In: *IEEE Computer Graphics and Applications (CGA)* 21.5 (2001), pp. 34–41.

[Rei+02] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. “Photographic Tone Reproduction for Digital Images.” In: *ACM Transactions on Graphics (TOG)* 21.3 (2002), pp. 267–276.

[Rei+06] E. Reinhard, P. Debevec, G. Ward, K. Myszkowski, H. Seetzen, D. Hess, G. McTaggart, and H. Zargarpour. “High Dynamic Range Imaging: Theory and Practice.” In: *ACM SIGGRAPH Courses*. 2006, pp. 1–320.

[Rei+10] E. Reinhard, W. Heidrich, P. Debevec, S. Pattanaik, G. Ward, and K. Myszkowski. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. 2nd. Morgan Kaufmann, 2010.

[RG15] K. Rohmer and T. Grosch. “Tiled Frustum Culling for Differential Rendering on Mobile Devices.” In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2015, pp. 37–42.

[RGS09] T. Ritschel, T. Grosch, and H.-P. Seidel. “Approximating Dynamic Global Illumination in Image Space.” In: *Proc. Symposium on Interactive 3D Graphics and Games (I3D)*. 2009, pp. 75–82.

[RH01a] R. Ramamoorthi and P. Hanrahan. “A Signal-processing Framework for Inverse Rendering.” In: *Proceedings of SIGGRAPH ’01*. 2001, pp. 117–128.

[RH01b] R. Ramamoorthi and P. Hanrahan. “An Efficient Representation for Irradiance Environment Maps.” In: *Proceedings of SIGGRAPH ’01*. 2001, pp. 497–500.

[RHL02] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. “Real-time 3D Model Acquisition.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 21.3 (2002), pp. 438–446.

[Ric+16] T. Richter-Trummer, D. Kalkofen, J. Park, and D. Schmalstieg. “Instant Mixed Reality Lighting from Casual Scanning.” In: *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2016, pp. 27–36.

[Ric70] J. P. Richter. *The Notebooks of Leonardo da Vinci: Compiled and Edited from the Original Manuscripts Vol. 1.* Vol. 1. New York: Dover Publications, Inc., 1970.

[Rit+08] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. "Imperfect Shadow Maps for Efficient Computation of Indirect Illumination." In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 27.5 (2008), 129:1–129:8.

[Rit+09] T. Ritschel, T. Engelhardt, T. Grosch, H.-P. Seidel, J. Kautz, and C. Dachsbacher. "Micro-Rendering for Scalable, Parallel Final Gathering." In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 28.5 (2009), 132:1–132:8.

[Rit+12] T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz. "The State of the Art in Interactive Global Illumination." In: *Computer Graphics Forum (CGF)* 31.1 (2012), pp. 160–188.

[RJG17] K. Rohmer, J. Jendersie, and T. Grosch. "Natural Environment Illumination: Coherent Interactive Augmented Reality for Mobile and non-Mobile Devices." In: *IEEE Transactions on Visualization and Computer Graphics (Proc. ISMAR)* 23.11 (2017), pp. 2474–2484.

[RL00] S. Rusinkiewicz and M. Levoy. "QSplat: A Multiresolution Point Rendering System for Large Meshes." In: *Proceedings of SIGGRAPH '00*. 2000, pp. 343–352.

[Roh+14] K. Rohmer, W. Büschel, R. Dachselt, and T. Grosch. "Interactive Near-Field Illumination for Photorealistic Augmented Reality on Mobile Devices." In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2014, pp. 29–38.

[Roh+15] K. Rohmer, W. Büschel, R. Dachselt, and T. Grosch. "Interactive Near-Field Illumination for Photorealistic Augmented Reality with Varying Materials on Mobile Devices." In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 21.12 (2015), pp. 1349–1362.

[Ros+95] E. Rose, D. Breen, K. H. Ahlers, C. Crampton, M. Tuceryan, R. Whitaker, and D. Greer. "Annotating Real-World Objects Using Augmented Reality." In: *Proc. Computer Graphics International*. 1995, pp. 357–370.

[RPG16] J. Riviere, P. Peers, and A. Ghosh. "Mobile Surface Reflectometry." In: *Computer Graphics Forum (CGF)* 35.1 (2016), pp. 191–202.

[RSS08] M. Rufli, D. Scaramuzza, and R. Siegwart. "Automatic Detection of Checkerboards on Blurred and Distorted Images." In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2008, pp. 3121–3126.

[Sch+14] C. Schwartz, R. Sarlette, M. Weinmann, M. Rump, and R. Klein. "Design and implementation of practical bidirectional texture function measurement devices focusing on the developments at the University of Bonn." In: *Sensors* 14.5 (2014), pp. 7753–7819.

[Sch+17] T. Schöps, M. R. Oswald, P. Speciale, S. Yang, and M. Pollefeys. "Real-Time View Correction for Mobile Devices." In: *IEEE Transactions on Visualization and Computer Graphics (Proc. ISMAR)* 23.11 (2017), pp. 2455–2462.

[Sch+93] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenberg. "Painting with Light." In: *Proceedings of SIGGRAPH '93*. 1993, pp. 143–146.

[Sch94] C. Schlick. “An Inexpensive BRDF Model for Physically-based Rendering.” In: *Computer Graphics Forum (CGF)* 13.3 (1994), pp. 233–246.

[Seg+06] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. “Bidirectional Instant Radiosity.” In: *Proc. Eurographics Symposium on Rendering (EGSR)*. 2006, pp. 389–397.

[SH16] D. Schmalstieg and T. Höllerer. *Augmented Reality: Principles and Practice*. 1st. Addison-Wesley Professional, 2016.

[Sin+08] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. “Interactive 3D Architectural Modeling from Unordered Photo Collections.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 27.5 (2008), 159:1–159:10.

[SIP07] B. Segovia, J. C. Iehl, and B. Péroche. “Metropolis Instant Radiosity.” In: *Computer Graphics Forum (CGF)* 26.3 (2007), pp. 425–434.

[SKS02] P.-P. Sloan, J. Kautz, and J. Snyder. “Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 21.3 (2002), pp. 527–536.

[Slo08] P.-P. Sloan. “Stupid Spherical Harmonics (SH) Tricks.” In: *Game Developers Conference (GDC)*. 2008, pp. 1–42.

[SM99] P. Sturm and S. Maybank. “A Method for Interactive 3D Reconstruction of Piecewise Planar Objects from Single Images.” In: *Proc. British Machine Vision Conference (BMVC)*. 1999, pp. 265–274.

[SMS06a] D. Scaramuzza, A. Martinelli, and R. Siegwart. “A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion.” In: *Proc. IEEE International Conference of Vision Systems (ICVS)*. 2006, 45:1–45:8.

[SMS06b] D. Scaramuzza, A. Martinelli, and R. Siegwart. “A Toolbox for Easily Calibrating Omnidirectional Cameras.” In: *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*. 2006, pp. 5695–5701.

[Son15] C. Sonderfeld. “Using a Live Reconstruction of the Real World for Global Illumination in Augmented Reality.” PhD thesis. Otto-von-Guericke-Universität Magdeburg, Germany, 2015.

[SRK13] C. Schwartz, R. Ruiters, and R. Klein. “Level-of-Detail Streaming and Rendering using Bidirectional Sparse Virtual Texture Functions.” In: *Computer Graphics Forum (Proc. Pacific Graphics)* 32.7 (2013), pp. 345–354.

[SSI03] I. Sato, Y. Sato, and K. Ikeuchi. “Illumination from shadows.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 25.3 (2003), pp. 290–300.

[SSI99] I. Sato, Y. Sato, and K. Ikeuchi. “Acquiring a Radiance Distribution to Superimpose Virtual Objects onto a Real Scene.” In: *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 5.1 (1999), pp. 1–12.

[ST90] T. Saito and T. Takahashi. “Comprehensible Rendering of 3-D Shapes.” In: *Computer Graphics (Proc. SIGGRAPH)* 24.4 (1990), pp. 197–206.

[Sta+96] A. State, G. Hirota, D. T. Chen, W. F. Garrett, and M. A. Livingston. “Superior Augmented Reality Registration by Integrating Landmark

Tracking and Magnetic Tracking." In: *Proceedings of SIGGRAPH '96*. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 429–438.

[Ste14] A. Stein. "Sensorbasierte Beleuchtung unter Berücksichtigung des Umgebungslichts für Augmented Reality Anwendungen auf mobilen Geräten." Bachelor's thesis. Otto-von-Guericke-Universität Magdeburg, Germany, 2014.

[SWI97] Y. Sato, M. D. Wheeler, and K. Ikeuchi. "Object Shape and Reflectance Modeling from Observation." In: *Proceedings of SIGGRAPH '97*. 1997, pp. 379–387.

[Sze11] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2011.

[Szi+05] L. Szirmay-Kalos, B. Aszódi, I. Lazányi, and M. Premecz. "Approximate Ray-Tracing on the GPU with Distance Impostors." In: *Computer Graphics Forum (CGF)* 24.3 (2005), pp. 695–704.

[Tan+13] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. "Live Metric 3D Reconstruction on Mobile Phones." In: *Proc. International Conference on Computer Vision (ICCV)*. 2013, pp. 65–72.

[Toc+11] M. D. Tocci, C. Kiser, N. Tocci, and P. Sen. "A Versatile HDR Video Production System." In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 30.4 (2011), 41:1–41:10.

[Tri+99] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. "Bundle Adjustment - A Modern Synthesis." In: *Proc. International Workshop on Vision Algorithms: Theory and Practice (ICCV)*. ICCV '99. 1999, pp. 298–372.

[TS67] K. E. Torrance and E. M. Sparrow. "Theory for off-specular reflection from roughened surfaces." In: *Journal of the Optical Society of America* 57.9 (1967), pp. 1105–1114.

[UG07] J. Unger and S. Gustavson. "An Optical System for Single-Image Environment Maps." In: *ACM SIGGRAPH Posters*. 2007, p. 185.

[UGY07] J. Unger, S. Gustavson, and A. Ynnerman. "Spatially varying image based lighting by light probe sequences." In: *The Visual Computer* 23.7 (2007), pp. 453–465.

[Ung+03] J. Unger, A. Wenger, T. Hawkins, A. Gardner, and P. Debevec. "Capturing and Rendering with Incident Light Fields." In: *Proc. Eurographics Symposium on Rendering (EGSR)*. 2003, pp. 141–149.

[Ung+13] J. Unger, J. Kronander, P. Larsson, S. Gustavson, J. Löw, and A. Ynnerman. "Spatially Varying Image Based Lighting using HDR-video." In: *Computers & Graphics* 37.7 (2013), pp. 923–934.

[Ung08] J. Unger. "Incident Light Fields." PhD thesis. Linköping University, Sweden, 2008.

[Vea97] E. Veach. "Robust monte carlo methods for light transport simulation." PhD thesis. Stanford University, 1997.

[VG95] E. Veach and L. Guibas. "Bidirectional Estimators for Light Transport." In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 1995, pp. 145–167.

[VG97] E. Veach and L. J. Guibas. "Metropolis Light Transport." In: *Proceedings of SIGGRAPH '97*. 1997, pp. 65–76.

[Vie+96] J. Viega, M. J. Conway, G. Williams, and R. Pausch. “3D Magic Lenses.” In: *Proc. ACM Symposium on User Interface Software and Technology (UIST)*. 1996, pp. 51–58.

[Wal+05] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. “Lightcuts: A Scalable Approach to Illumination.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 24.3 (2005), pp. 1098–1107.

[Wal+07] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance. “Microfacet Models for Refraction Through Rough Surfaces.” In: *Proc. Eurographics Symposium on Rendering (EGSR)*. 2007, pp. 195–206.

[War92] G. J. Ward. “Measuring and Modeling Anisotropic Reflection.” In: *Computer Graphics (Proc. SIGGRAPH)* 26.2 (1992), pp. 265–272.

[WD01] J. Waese and P. Debevec. “A Real Time High Dynamic Range Light Probe.” In: *ACM SIGGRAPH Sketches*. 2001, p. 1.

[WEH89] J. R. Wallace, K. A. Elmquist, and E. A. Haines. “A Ray Tracing Algorithm for Progressive Radiosity.” In: *Computer Graphics (Proc. SIGGRAPH)* 23.3 (1989), pp. 315–324.

[Wei+09] T. Weise, T. Wismer, B. Leibe, and L. Van Gool. “In-hand Scanning with Online Loop Closure.” In: *International Conference on Computer Vision (ICCV) Workshops*. 2009, pp. 1630–1637.

[Wei+16] M. Weinmann, F. Langguth, M. Goesele, and R. Klein. “Advances in Geometry and Reflectance Acquisition.” In: *Eurographics Courses*. 2016, pp. 1–71.

[WFG92] L. R. Wanger, J. A. Ferwerda, and D. P. Greenberg. “Perceiving spatial relationships in computer-generated images.” In: *IEEE Computer Graphics and Applications (CGA)* 12.3 (1992), pp. 44–58.

[WH92] G. J. Ward and P. Heckbert. “Irradiance Gradients.” In: *Proc. Eurographics Workshop on Rendering (EGWR)*. 1992, pp. 85–98.

[Whe+12] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald. *Robust Tracking for Real-Time Dense RGB-D Mapping with Kintinuous*. Tech. rep. Massachusetts Institute of Technology, 2012, pp. 1–10.

[Whe+15] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. “ElasticFusion: Dense SLAM Without A Pose Graph.” In: *Robotics: Science and Systems (RSS)*. Vol. 11. 2015.

[Whi80] T. Whitted. “An Improved Illumination Model for Shaded Display.” In: *Communications of the ACM* 23.6 (1980), pp. 343–349.

[Wil+05] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. “High Performance Imaging Using Large Camera Arrays.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 24.3 (2005), pp. 765–776.

[WK15] M. Weinmann and R. Klein. “Advances in Geometry and Reflectance Acquisition.” In: *ACM SIGGRAPH Asia Courses*. 2015, 1:1–1:71.

[Wol15] S. Wolligandt. “Bildbasierte Optimierung von optischen Trackingsystemen in Augmented Reality-Anwendungen.” Bachelor’s thesis. Otto-von-Guericke-Universität Magdeburg, Germany, 2015.

[Xia+16] R. Xia, Y. Dong, P. Peers, and X. Tong. “Recovering Shape and Spatially-Varying Surface Reflectance under Unknown Illumination.” In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), 187:1–187:12.

[XM06] X. Xiao and L. Ma. “Color Transfer in Correlated Color Space.” In: *Proc. ACM International Conference on Virtual Reality Continuum and Its Applications (VRCIA)*. 2006, pp. 305–309.

[Yag+11] A. Yagi, M. Imura, Y. Kuroda, and O. Oshiro. “360degreee Fog Projection Interactive Display.” In: *ACM SIGGRAPH Asia Emerging Technologies*. 2011, 19:1–19:1.

[Yan+10] J. C. Yang, J. Hensley, H. Grün, and N. Thibieroz. “Real-time Concurrent Linked List Construction on the GPU.” In: *Computer Graphics Forum (Proc. EGSR)*. 2010, pp. 1297–1304.

[YKK12] Y. Yao, H. Kawamura, and A. Kojima. “Shading Derivation from an Unspecified Object for Augmented Reality.” In: *Proc. International Conference on Pattern Recognition (ICPR)*. 2012, pp. 57–60.

[Yu+99] Y. Yu, P. Debevec, J. Malik, and T. Hawkins. “Inverse Global Illumination: Recovering Reflectance Models of Real Scenes from Photographs.” In: *Proceedings of SIGGRAPH ’99*. 1999, pp. 215–224.

[ZCC16] E. Zhang, M. F. Cohen, and B. Curless. “Emptying, Refurnishing, and Relighting Indoor Spaces.” In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 35.6 (2016), pp. 1–14.

[Zha+02] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz. “Single-view modelling of free-form scenes.” In: *Journal of Visualization and Computer Animation* 13.4 (2002), pp. 225–235.

[Zha00] Z. Zhang. “A Flexible New Technique for Camera Calibration.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334.

[Zwi+01] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. “Surface Splatting.” In: *Proceedings of SIGGRAPH ’01*. 2001, pp. 371–378.